МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.04.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Информационные системы и технологии корпоративного управления
(направленность (профиль)/специализация

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: «Аналитический инструментарий поддержки программного продукта на основе методов полнотекстового поиска»

Студент Е.К. Токмаков _____
(И.О. Фамилия) (личная подпись)

Научный руководитель А.В. Очеповский _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д.т.н., доцент, С.В. Мкртычев _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
«_____» _____20_____г. (личная подпись)

Тольятти 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..... 3
1 Теоретико-методологические основы технической поддержки программного обеспечения..... 8
1.1 Проблемы технической поддержки программного обеспечения 8
1.2 Процесс обеспечения гарантии качества программных средств 12
1.3 Процесс решения проблем в программных средствах 16
2 Анализ существующих методов и подходов к решению проблемы инструментария поддержки программных средств..... 23
2.1 Сравнение современных продуктов работы с данными 24
2.2 Моделирование тестовой среды 27
3 Представление авторского решения поставленной в исследовании проблемы34
3.1 Проектирование аналитического инструмента 34
3.2 Реализация аналитического инструмента 40
4 Представление экспериментальных и расчетных результатов апробации 52
4.1 Проверка соответствия решения заданным требованиям 52
4.2 Оценка применимости решения в процессе сопровождения 63
ЗАКЛЮЧЕНИЕ 67
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ..... 70

ВВЕДЕНИЕ

С каждым годом информационные технологии все обширнее охватывают быт человека, проникая в новые сферы, укрепляясь и совершенствуясь в уже достигнутых вершинах. Там, где технологии ранее не использовались – они активно разрабатываются и внедряются. К ярким примерам таких технологий можно отнести:

- 3D-печать на принтерах материалами: пластик, металл;
- «умные» дома, «умные» города, - жилая местность, оснащенная автоматизированными технологиями для поддержания жизнеобеспечивающей инфраструктуры при отсутствии человеческого контроля и управления;
- технология blockchain – развитие крипто-валюты, которая может заменить традиционные товарно-денежные отношения.

В других областях, где, казалось бы, за последние годы развитие уже перешагнуло через все вообразимые границы, технологии тоже продолжают развиваться и внедряться в человеческий быт:

- 5G – новое поколение сетей передачи данных, превышающее в 10-15 раз предыдущее поколение 4G;
- централизация государственных социальных услуг населению в России в единую систему «ГосУслуги»;
- «андроидизация» услуг: с учетом массовости и доступности использования смартфонов и мобильных сетей, быстрыми темпами растет количество компаний, имеющих собственные андроид-приложения для предоставления различного вида услуг.

Каждая технология помимо технической составляющей (оборудование) подразумевает также и программную составляющую, которая имеет непрерывный цикл развития и перерождения в течение всего периода конкурентоспособности

продукта. Несмотря на развитие способов и методов ведения разработки, проектирования и реализации, любому продукту необходима поддержка и ресурсы, осуществляющие поддержку, выделяются в каждой компании, занимающейся разработкой программного обеспечения. Длительность поддержки программного продукта напрямую зависит от его востребованности. Но даже на продукты с низким уровнем популярности отсутствие либо низкое качество поддержки может сыграть крайне негативную роль: в случае с заказной разработкой компания может лишиться заказчика и получить негативную огласку, а если приложение создано не на заказ, то на конкурентном рынке аналогичных приложений спрос на него упадет.

Важность процесса поддержки программных средств, коммерческая обоснованность качественного предоставления услуг по сопровождению программных продуктов, необходимость в непрерывном совершенствовании организационных процессов жизненного цикла программного обеспечения обусловили выбор темы исследования, определили цель, объект и предмет исследования.

Актуальность диссертационного исследования определила проблема некачественной поддержки программных средств как в России, так и за рубежом.

Объект исследования - процесс сопровождения программных средств.

Предмет исследования - аналитический инструмент поддержки программного продукта.

Цель исследования – теоретическое обоснование и практическая реализация модели аналитического инструмента поддержки программного продукта на основе системы полнотекстового поиска для улучшения процесса работы с централизацией и анализом данных работы продукта, повышения качества услуг сопровождения.

В процессе исследования автором выдвинута **рабочая гипотеза** о том, что процесс сопровождения программных средств может быть улучшен, если:

- выявлены и сформулированы проблемы осуществления поддержки;
- определены функциональные нужды процесса поддержки и разработан аналитический инструмент для покрытия недостающего функционала.

Цель исследования и сформулированная рабочая гипотеза обусловили следующие **задачи:**

1. Исследовать процесс сопровождения программного обеспечения.
2. Смоделировать имитацию процесса с применением доработанного продукта программного обеспечения с открытым исходным кодом в качестве инструмента поддержки программного обеспечения.
3. Протестировать полученный продукт с использованием сгенерированных тестовых данных.
4. Определить оправданность использования продуктов с открытым исходным кодом в качестве инструмента поддержки программного обеспечения.

Теоретической и методологической основой магистерского исследования являются методологические основы сопровождения программных систем М. Аншиной, К.А. Бутузова, П. Джалота, С. Орлик, В.П. Селезнева, В.Л. Хан, Б. Висванатана, П. Дюваля, П.Д. Виндлея; анализ достоинств применения продуктов с открытым исходным кодом Н. Барановой; подходы к оценке деятельности на основе метрических показателей О. Кулагина, В.В. Липаева, Б. Фрэнкса; статистические исследования затрат рабочего времени сотрудников А. Бочкина, Д. Брауна; методологические основы разработки программных средств А.К. Дадыкина, А.А. Ермолаева, Г. Ефимова, Н.Ю. Иванова, В.Г. Маняхиной, В.В. Липаева, В.П. Селезнева, А.А. Смирнова; подходы к проведению исследовательской работы Ю.В. Казаковой; методы мониторинга информационных систем М.Г. Дубровина, И.Н. Глухих, С.А. Славкова; сравнительный анализ полнотекстовых поисковых систем А. Клименко.

В процессе исследования использованы следующие **методы**:

- *теоретические*: анализ научно-технической литературы, нормативной документации;
- *эмпирические*: наблюдение процессов поддержки на основе опыта работы в аналогичной сфере в компании по разработке программного обеспечения, моделирование процесса поддержки.

Научная новизна. Как результат исследования, разработан аналитический инструмент поддержки на основе системы полнотекстового поиска, позволяющий оптимизировать процессы работы специалистов сопровождения программных средств и повысить качество предоставляемых услуг.

Теоретическая значимость диссертационного исследования состоит в развитии информационных технологий в узкой сфере сопровождения программных средств, анализе процессов поддержки с целью улучшения, рассмотрении качества программного продукта на конечном этапе жизненного цикла на практической основе.

Практическая значимость диссертационного исследования состоит в разработке аналитического инструмента для применения в сфере поддержки программного обеспечения, развитии процессов сопровождения продуктов с использованием специализированных средств, функционально подготовленных для использования.

На защиту выносятся:

1. Модель аналитического инструмента поддержки на основе системы полнотекстового поиска.
2. Имплементация аналитического инструмента поддержки, готового к использованию на практике.
3. Апробация разработанного продукта, анализ применимости данного вида решения для улучшения процессов сопровождения программных средств и повышения качества предоставляемых услуг.

Апробация. Тестирование полученного продукта с использованием критериев оценки следующих групп проверки:

- консистентность данных;
- покрытие функциональных требований.

В результате апробации имплементация аналитического инструмента подтвердила правильность реализации и подтвердила возможность применения полученного продукта в сфере поддержки программных средств.

Диссертация состоит из введения, четырех глав, заключения, списка литературы и приложений. Работа изложена на 75 страницах, содержит 26 рисунков, 5 таблиц.

В первой главе рассматриваются процессы сопровождения программных средств, анализируются нужды инженеров поддержки, конкретизируются проблемы и определяются функциональные требования к аналитическому инструменту.

Во второй главе сравниваются современные популярные системы полнотекстового поиска, осуществляется выбор системы для реализации аналитического инструмента, разрабатывается тестовая среда, структура тестовых данных и методов их генерации, составляется набор тестов для проверки реализации.

В третьей главе проектируется модель инструмента и его компонентов, алгоритмов их работы, выполняется имплементация аналитического инструмента, включающая настройку и конфигурирование, разработку и отладку компонентов.

В четвертой главе выполняется тестирование аналитического инструмента, полученный продукт проверяется на соответствие обозначенным функциональным требованиям, оценивается его применимость.

В заключении подводятся итоги выполненной работы.

1 ТЕОРЕТИКО-МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1 Проблемы технической поддержки программного обеспечения

Важность поддержки программного обеспечения обусловлена не только удержанием на рынке продукта разработки, но и своего рода возможностью обратной связи функционирования продукта в реальной работе - информированием о качестве реализации и сбором статистических данных для развития новых версий. Если взять как пример рынок мобильных приложений Play Market самой распространенной мобильной операционной системы Android и посмотреть статистику количества зарегистрированных приложений, доступных для загрузки за 2018 год, то окажется что в пиковый период, который пришелся на март месяц, общее количество приложений превысило 3,6 миллионов [31]. Это показывает очень высокий уровень конкуренции: даже если учесть, что программные продукты можно разделить на 58 тематический категорий [32] и из каждой категории выделить существенные отличительные черты для каждой подгруппы, конкуренция все равно будет иметь место. Захват рынка новым функционалом или решительно уникальными технологиями стал редкостью, наиболее ценным для разработчиков стало удержание пользователей.

Качество продукта, возможности персонализации, своевременные обновления и обратная связь с пользователем, подразумевающая учет пожеланий для развития, стали играть большую роль, сам факт создания успешного приложения уже перестал быть залогом успеха. Учитывая реактивность конкурентной среды, уникальность функционала, недавно представленного на рынке, может быстро потеряться в аналогичных приложениях, всегда готовых на скорую руку перенять

идею и выполнить собственную реализацию, отнимая прибыль у новатора. Низкий уровень поддержки и отсутствие команды за нее ответственной могут стать определяющими факторами в дальнейшей судьбе программного продукта.

Выделение ресурсов для обеспечения поддержки программного продукта возможно только для компаний, ведущей несколько проектов, в том числе успешных, которые являются основой для существования. Но даже в компаниях крупнее среднестатистических, неважно, занимается ли она разработкой мобильных приложений или программного обеспечения для роутеров, поддержка всегда будет являться спорной затратой, на которую будет выделяться минимум ресурсов: как технических, так и людских. В данном аспекте не стоит забывать, что если для однопроектной компании двух-трех единомышленников, занимающихся разработкой софта, одного человеко-ресурса самого по себе будет вполне достаточно, то в случае с многопроектным потоком только людей будет недостаточно.

Выполнение контрактных обязательств поддержки – первоочередная задача к исполнению на любом проекте, осуществляющем поддержку программного продукта. Качество выполнения напрямую влияет на прибыль компании-разработчика, замеряется условленными метриками. Для расчета метрик и поддержания продукта на должном уровне качества с баз данных серверов регулярно снимаются агрегированные данные для построения статистики и анализа трендов, выявления пиков и предсказания проблем. Такая нагрузка на базы данных может выполнять в отдельных случаях полное сканирование табличных записей и значительно повышать скорость потока чтения с жестких дисков, однако сама по себе не несет никакого функционального смысла и может влиять на бизнес-процессы. Возможны снятия отчетов в нерабочее время, но это усложняет процесс своевременного получения актуальной информации. Временные затраты на ручную подготовку отчетов помимо использования технологических ресурсов также сильно сокращают полезное время, которое

сотрудник имеет в своем распоряжении [7]. На рисунке 1 автор приводит пример статистики расходуемого времени сотрудника на отчетность в течение рабочей недели на основе собственного опыта работы в компании разработки программного обеспечения. Диаграмма показывает, что в процентном соотношении в среднем в неделю 15% времени потрачено на подготовку отчетности по заказам.

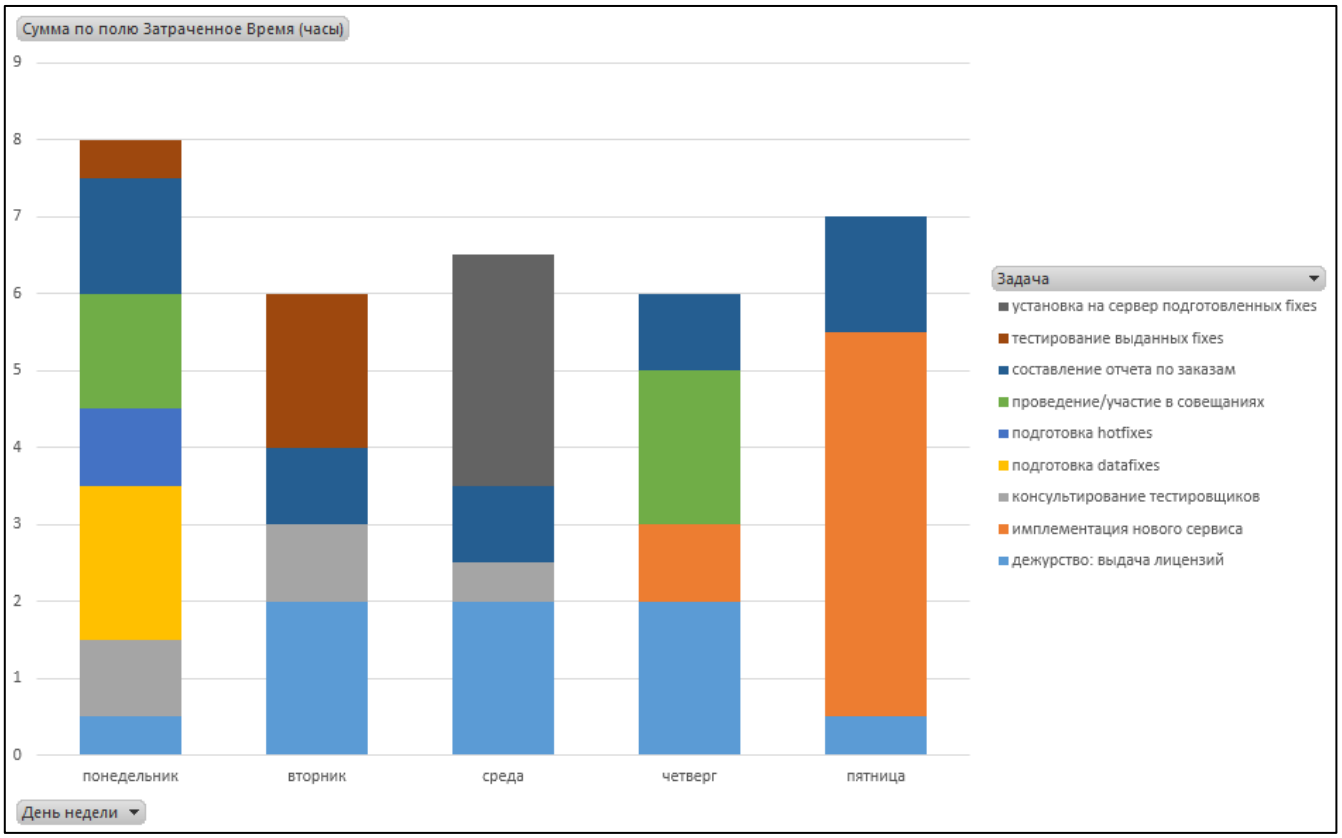

Рисунок 1 - Статистика расходуемого времени выполнения рабочих заданий за неделю

Автор выделяет следующие проблемы в поддержке программных средств:

- отсутствие инструментов централизованного доступа к информации для анализа проблем;
- нефункциональная нагрузка на базы данных для сбора статистики;

- высокий процент времени, затрачиваемого сотрудниками на ведение отчетности;
- ограниченность ресурсов, выделяемых на поддержку, что усложняет улучшение процессов.

Решением данной проблемы по предложению автора является аналитический инструмент поддержки программных средств. Для исследования технологических нужд процесса поддержки и, как следствие, уточнения функциональных особенностей инструмента, автор предлагает рассмотреть, что представляет собой сам процесс поддержки и какие стандарты описывают его составляющие.

Поддержка программного продукта – это не технология, а процесс или, вернее сказать, этап жизненного цикла продукта. Данный этап не имеет однозначной структуры или неизменных алгоритмов существования, однако существует общее описание поддержки программного продукта как деятельности в международных стандартах с указанием ключевых моментов и обязательных требований к организации. Также возможно обратиться к помощи различных рекомендаций как от отдельных опытных специалистов в этой области, так и от компаний, имеющих в своем послужном списке значительный стаж работы в сфере поддержки и сопровождения информационных технологий и программного обеспечения в частности.

Национальный стандарт Российской Федерации ГОСТ Р ИСО/МЭК 12207—2010 описывает жизненный цикл программного обеспечения [1]. Все виды деятельности, которые могут реализовываться в той или иной мере и входить в жизненный цикл программных систем, разделены в стандарте 12207 на группы процессов.

Группы процессов жизненного цикла программного обеспечения согласно стандарту ГОСТ Р ИСО/МЭК 12207—2010:

1. Процессы соглашения (2 процесса).
2. Процессы организационного обеспечения проекта (5 процессов).

3. Процессы проекта (7 процессов).
4. Технические процессы (11 процессов).
5. Процессы реализации программных средств (7 процессов).
6. Процессы поддержки программных средств (8 процессов).
7. Процессы повторного применения программных средств (3 процесса).

В рамках исследовательской работы автор фокусирует свое внимание на группе процессов поддержки программных средств.

Группа процессов поддержки программных средств, согласно стандарту, включает в себя следующие процессы:

1. Процесс менеджмента документации программных средств.
2. Процесс менеджмента конфигурации программных средств.
3. Процесс обеспечения гарантии качества программных средств.
4. Процесс верификации программных средств.
5. Процесс валидации программных средств.
6. Процесс ревизии программных средств.
7. Процесс аудита программных средств.
8. Процесс решения проблем в программных средствах.

Наиболее интересными с точки зрения исследовательской работы являются следующие процессы: обеспечения гарантии качества программных средств, решения проблем в программных средствах.

Автор предлагает рассмотреть подробнее выделенные процессы поддержки.

1.2 Процесс обеспечения гарантии качества программных средств

Целью данного процесса является обеспечение гарантии качества программного средства в соответствии с требованиями, которые оговорены между разработчиком и заказчиком на основе соглашения сторон. В требованиях фиксируются показатели, по которым принято характеризировать качество

продукта и на основе которых можно судить о выполнении или невыполнении гарантийных обязательств. Также в соглашение включается ответственность сторон, в частности, применительно к гарантии качества, штрафная система и условия расторжения соглашения. Гарантия качества соблюдается при условии постановки и соблюдения стратегии обеспечения гарантии качества, в которую входят как технические, так и организационные ресурсы компании-разработчика.

Процесс обеспечения гарантии качества программных средств тесно пересекается с процессом валидации программных средств. В одну из задач исследовательской работы входит поиск способа предоставления технического решения для проверки обеспечения гарантии, а точнее – средства для валидации гарантийных обязательств.

Реализация данного процесса, помимо определения показателей, состоит также и в установке способов и технологий расчета показателей. Технологии могут быть предложены как со стороны компании-заказчика, так и со стороны компании-разработчика. В задачи исследовательской работы входит рассмотрение средств реализации расчета данных показателей, а также ведения мониторинга трендов изменения показателей с течением времени для возможности проведения аналитики.

Результаты должны быть доступными всем участникам соглашения. При этом все выявленные проблемы (на основе несоответствия достигнутых показателей и требуемых значений) должны быть устранены. Такие требования предписывают максимально детально определять в рамках соглашения что и какими способами следует замерять. Однозначность толкования способов должна быть учтена при создании соглашения между сторонами.

Построенный процесс обеспечения гарантии качества имеет однозначные решения следующих вопросов:

1. Что необходимо проверять.
2. Как необходимо проверять.

3. С помощью чего необходимо проверять.

Объекты проверки варьируются в зависимости от проекта: все зависит от бизнес-данных и общей цели программного решения. Также не стоит забывать, что валидация решения применима не только к функциональным требованиям, но и к техническим. Отказоустойчивость работы жесткого диска в серверном массиве, к примеру, является характеристикой работы как сервера, так и работающего на нем программного обеспечения в целом.

Процесс для определения, описания, выбора, применения и совершенствования методик оценки программного средства является частью сопровождения. При этом сопроводитель должен собрать, проанализировать и интерпретировать необходимые данные о работе продукта. Это подразумевает реализацию способов и методов сбора информации о работе программного средства: как исторической, так и текущей, актуальной.

Ключевую роль в процессе обеспечения качества играет своевременное уведомление о возможных угрозах работы программного средства для принятия мер по обслуживанию системы и/или оборудования. Периодические проверки, сконфигурированные для различных частей системы, имеют уровни приоритетов и в соответствии с наиболее высокими приоритетами система обслуживания должна уметь оповещать персонал поддержки о возможных угрозах и сбоях, сработавших триггерах и случившихся событиях [11]. Наиболее распространенным способом уведомления в проактивном мониторинге работы системы является почтовое уведомление с краткой или развернутой информацией о месте возникновения события, ключевых показателей, спровоцировавших отправление уведомления, допустимую норму показателей. Почтовое уведомление также может содержать в себе и визуальную информацию со статистикой проверяемых показателей за последний временной промежуток.

Примеры объектов и предметов проверок с указанием возможных методов проверки указаны в таблице 1.

Таблица 1 - Примеры объектов проверки программного продукта и возможные методы их проверки

Объект	Предмет проверки	Способ проверки
1. Бизнес-данные	Процессы создания/исполнения бизнес-сущностей	Расчет показателей на основе калькуляционных запросов к базе данных
2. Серверное оборудование	Заполнение выделенной памяти жестких дисков	Проверка свободного места в разделах жестких дисков на уровне операционной системы
	Загрузка сети трафиком	Проверка загрузки сети на уровне операционной системы
	Загрузка оперативной памяти, кэширование	Проверка состояния оперативной памяти
	Загрузка дисков чтением/записью данных	Проверка ввода-вывода физических дисков
	Отсутствие ошибок в работе серверной операционной системы	Проверка наличия ошибок в логах операционной системы
3. Информационная система (решение)	Доступность информационной системы	Расчет показателей на основе времени недоступности системы (устранение сбоев в работе, плановое обслуживание)
	Время отклика	Расчет времени загрузки окна пользователя информационной системы
	Время выполнения функциональных задач	Расчет времени выполнения задачи информационной системой
	Отсутствие ошибок в работе информационной системы (решения)	Проверка наличия ошибок в логах информационной системы

Способы проверки программных средств, приведенные в таблице 1, позволяют вывести следующие функциональные требования к аналитическому инструменту для организации процесса мониторинга информационной системы:

1. Возможности сбора информации с различных источников данных (базы данных, текстовые файлы).
2. Доступность фильтров (и парсинга, применительно к текстовым файлам) для сбора только необходимой для анализа информации.
3. Наличие способа уведомлений с использованием почтовой рассылки.
4. Возможности планирования периодических проверок.
5. Возможности калькуляции необходимых показателей по заданным алгоритмам расчета.

1.3 Процесс решения проблем в программных средствах

Цель процесса – обеспечение гарантии в учете, обработке и предоставлении своевременных решений проблемам, возникшим в ходе эксплуатации программного продукта. Основой определения стратегии сопровождения является концепция сопровождения.

Концепция сопровождения – это отражение:

- области сопровождения;
- практического применения процесса сопровождения;
- определение ответственных за сопровождение лиц;
- стоимости сопровождения.

Определение области сопровождения явно указывает на характеристику и возможные места возникновения проблем, которые покрываются процессом поддержки. Данное условие предъявляет требования к возможности ограничения информации, которая нуждается в анализе и мониторинге со стороны команды поддержки.

Стратегия работы с проблемами позволяет определить:

- что может быть названо проблемой и должно решаться компанией-разработчиком;
- как должна быть оформлена проблема для обеспечения возможности ее обработки;
- какие системы будут использованы для работы с проблемами и мониторинга их выполнения;
- как будет устроена их приоритезация;
- с какой частотой будут выполнены выдачи решения по заведенным проблемам;
- какие виды проблем могут быть учтены;
- какие сроки будут являться определяющими для каждого из видов проблем в соответствии с указанным приоритетом.

В соответствии с процессом также определяется порядок и частота выдачи отчетов, их структура, организация встреч для обсуждения проблем. Однако к данному процессу можно отнести не только проблемы, обнаруженные заказчиком, но и те, которые были обнаружены мониторинговыми службами (если таковые используются), то есть в результате автоматических проверок.

Стандарт ГОСТ Р ИСО/МЭК 14764—2002 более подробно описывает процесс поддержки программного обеспечения [2]. Схематически процесс сопровождения показан на рисунке 2.

С точки зрения поддержки аналитическим средством самый трудоемкий этап для команды поддержки программного средства - анализ проблем и изменений.

Как правило, выявление коренных причин проблем продукта сводится к трем шагам:

1. Сбор необходимой информации.
2. Анализ данных.
3. Подготовка решения проблемы.

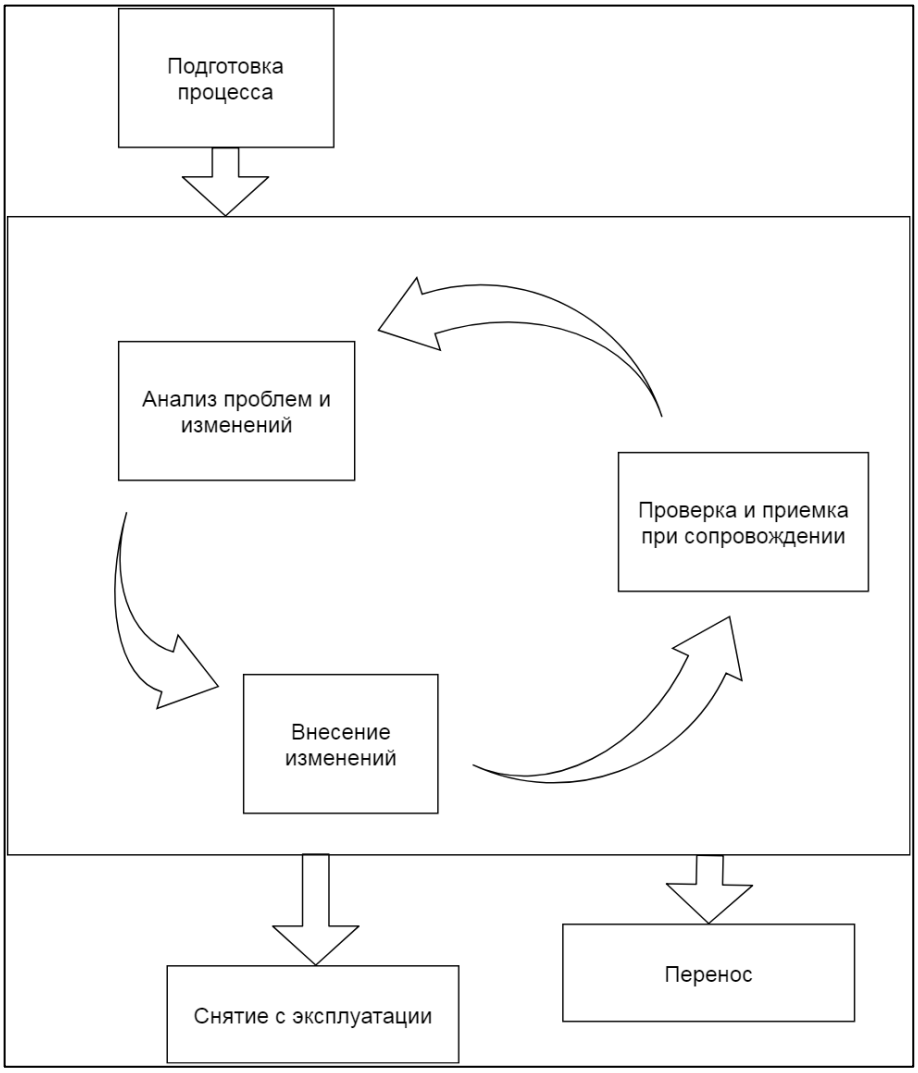

Рисунок 2 - Процесс сопровождения по стандарту ГОСТ Р ИСО/МЭК 14764—2002

На первый взгялд кажется, что первые два шага - наиболее нуждающиеся в аналитических технологиях и специальных программных продуктах. Однако на деле, ввиду элементарного отсутствия ресурсов для создания такого рода инструментов, сбор и анализ необходимой информации сводится к визуальному прочтению файлов логирования инженером поддержки, снятия слепков бизнес-данных связанных бизнес-сущностей путем прямых запросов к базе данных либо обращению к информационной системе через пользовательский интерфейс.

В случае крупных проектов количество проблем может оказаться проблематичным для анализа. Попытка их анализа в порядке появления может привести к избыточной работе над схожими коренными причинами и неэффективности использования ресурсов поддержки. Этого можно избежать, внедрив способы типизации и группировки ошибок по заданным параметрам. Преимущество аналитической работы над группировкой схожих проблем показано на рисунке 3.

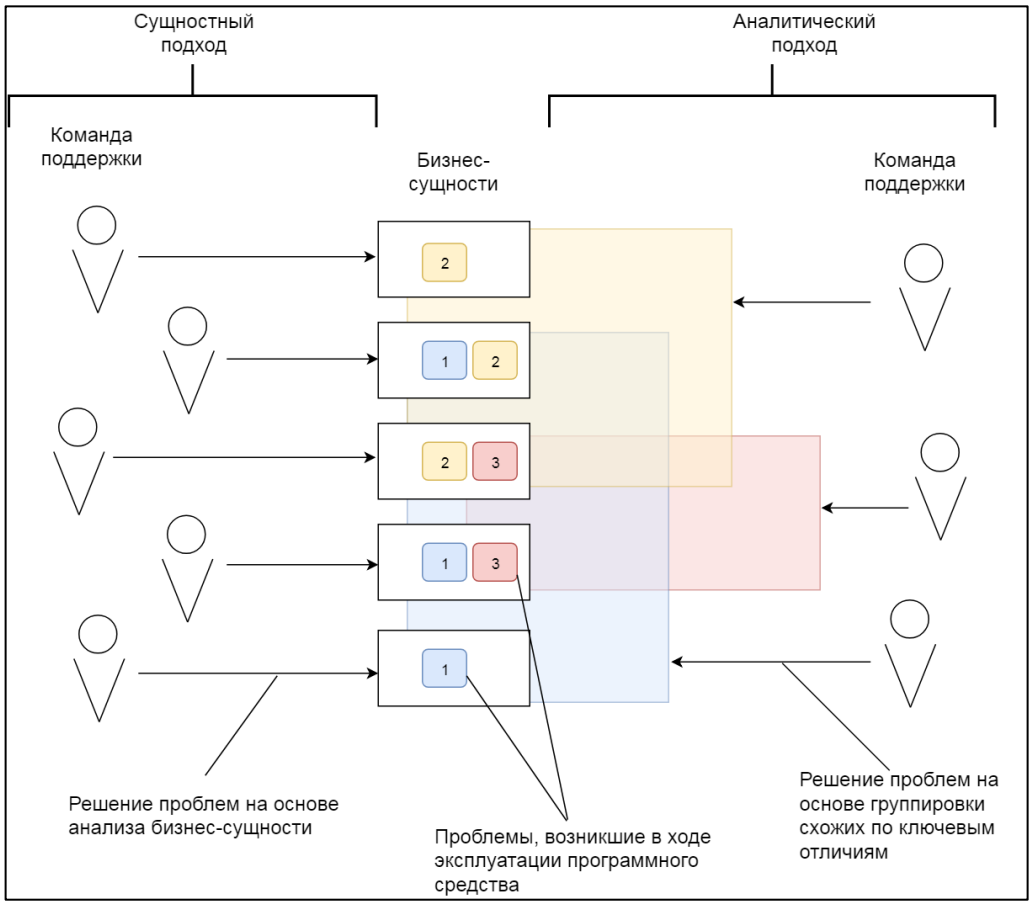

Рисунок 3 – Сущностный и аналитический подходы при решении проблем

Сконфигурированные группы проблем могут быть использованы в построении статистических данных по самым распространенным проблемам и улучшить прогресс работы команды путем приоритезации разрешения проблем: от самых часто встречаемых к самым редко случающимся. Также на основе групп

можно выделить те проблемы, которые не входят в область обслуживания и могут быть проигнорированы.

Оптимизация данного процесса для повышения эффективности работы с проблемами подразумевает централизованный сбор необходимой информации и инструменты работы с ней, включая:

1. Фильтрацию по ключам.
2. Быстрый доступ.
3. Необходимые объемы собираемой информации из разных источников в полном доступе.
4. Связанность и целостность данных в системе по ключевым полям для удобства сбора информации.
5. Группировка однотипных проблем.

Также особое внимание в поддержке, как практически и в любой деятельности, уделяется отчетности как способу формирования представления о текущем положении дел. Отчетность используется на разных уровнях компании, однако наиболее сложными отчетами, ввиду их практической значимости, всегда будут отчеты наиболее приближенные к коммерческой выгоде. Такие отчеты, как правило, имеют сложную систему расчета, и отдельные сотрудники компании тратят около 6% своего времени на их подготовку ежедневно [7]. Формирование отчетов на автоматической основе может существенно снизить затраты времени сотрудников и освободить ресурсы.

Принимая во внимание необходимость создания отчетов, следующий функционал может быть с пользой использован для сбора статистической информации без участия сотрудников:

1. Построение агрегированных отчетов.
2. Визуальное представление данных в виде отчетов.
3. Отправка отчетов в виде электронного письма.

4. Планирование сборки отчетов в соответствии с сконфигурированным графиком.

На основе анализа составляющих процессов мониторинга и решения проблем информационной системы можно выделить следующие требования к аналитическому инструменту сопровождения:

1. Поддержка больших объемов данных с индексированным поиском по любым данным из доступных.
2. Поддержка различных источников данных для сбора информации (в том числе базы данных, текстовые файлы).
3. Минимизация нагрузки на базы данных при репликации данных.
4. Быстрый доступ к информации (менее 2 секунд для вывода информации по указанному запросу).
5. Гарантированность получения актуальной информации.
6. Возможности очистки данных по фильтрам.
7. API для работы с хранилищем данных для использования внешними системами.
8. Поддержка построения агрегированных отчетов.
9. Визуальное представление данных.
10. Возможность планирования запусков расчета показателей (для системы мониторинга и отчетности) с выходом в виде файлов отчета либо почтового сообщения.
11. Поддержка фильтрации входных данных (парсинг текстовых данных в том числе).

Данные требования будут использованы как основа для проектирования аналитического инструмента поддержки и итоговых выводов о возможности применения созданного решения в поддержке.

Для подготовки решения исследовательской работы на основе полученных требований выделены следующие задачи:

1. Проанализировать современные движки, базирующиеся на методах полнотекстового поиска.
2. Смоделировать тестовую среду для реализации решения в модели процесса поддержки программного средства.

Анализ процессов сопровождения программного обеспечения показал, что поддержка остро нуждается в технических средствах и аналитических инструментах. Опыт работы в компаниях разработки программного обеспечения позволяет сделать вывод о необходимости материально-технической подготовки для сотрудников отдела поддержки. Такого рода инструментарий существенно улучшит процессы работы отдела сопровождения, разгрузит персонал, занятый неквалифицированной работой, высвободит ресурсы для выполнения приоритетных задач. Государственные и международные стандарты не регламентируют качество, количество и типы используемого в поддержке инструментария, однако наличие такового является следствием цели сопровождения – поддержания работы программного средства на должном уровне требуемого качества.

2 АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ И ПОДХОДОВ К РЕШЕНИЮ ПРОБЛЕМЫ ИНСТРУМЕНТАРИЯ ПОДДЕРЖКИ ПРОГРАММНЫХ СРЕДСТВ

Первоначально автором было выполнено планирование работ и уточнение задач. Работы были поделены на три этапа: подготовка, наполнение, реализация. Схема плана работ показана на рисунке 4.

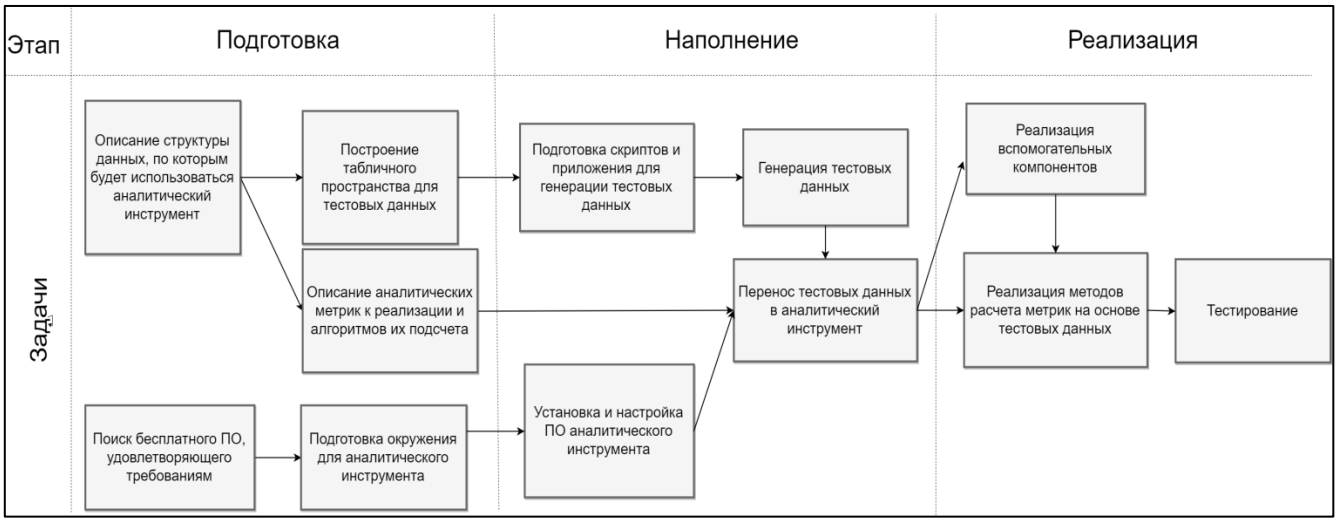

Рисунок 4 - Планирование работ по проектированию и реализации аналитического инструмента

На этапе подготовки автор выполняет сравнение популярных систем полнотекстового поиска с целью выбора системы для реализации модели процесса поддержки, описывает источники данных программного средства, как предмета поддержки, моделирует бизнес-данные, показательные метрики для анализа данных и способы их расчета и получения в виде отчетов для проведения проверки подготавливаемого аналитического инструмента.

2.1 Сравнение современных продуктов работы с данными

Для решения обозначенных проблем в процессе сопровождения программных средств и покрытия необходимых функциональных возможностей автором рассмотрены разработанные программные продукты, которые могут быть использованы в реализации аналитического инструмента.

На основе функциональных требований к инструменту, относящихся к способам хранения информации и методов работы с ними, автор определил технологию полнотекстового поиска как основу для реализации ввиду следующих его преимуществ:

- полная индексация данных для быстрого доступа;
- возможность работы с большими объемами неструктурированных данных.

Современные тренды развития дали широкое распространение программного обеспечения с открытым исходным кодом. Такие приложения свободны в использовании, бесплатны, и к тому же поддерживают любые доработки под специфические нужды.

Достоинства использования систем с открытым исходным кодом очевидны и актуальны для компании-разработчика любого уровня [8]:

- возможности доработки и персонализации;
- бесплатное использование;
- свободный выбор без привязки к маркетинговым кампаниям продукта.

Бесплатное использование может оказаться ключевым фактором при выборе продукта в условиях ограниченности ресурсов, что является одним из требований исследовательской работы. При подборе программных средств учитывались только продукты, использование которых не подразумевало платы.

На основе данных ранжирования популярных движков хранения и поиска данных [33] для рассмотрения программных решений автор выбрал следующие продукты полнотекстового поиска:

1. Solr.
2. Elastic Stack.
3. Sphinx.

Сравнительные данные продуктов представлены в таблице 2 с использованием нескольких источников [12,13,34,38].

Замеры индексации и скорости поиска, показывающей среднее время вывода ответа, выполнены при условиях:

- 3 слова для поиска;
- 5 потоков, задействованных в поиске;
- 1000 результатов;
- 80000 сущностей;
- общий объем данных ~330 мб;
- процессор Core 2 Duo 3 ГГц;
- выделенная оперативная память – 1 гб.

При небольшой разнице в скорости индексации новых данных и в скорости поиска по документам, все три продукта относительно схожи друг с другом и могут быть использованы для работы в качестве хранилищ с полнотекстовым методом поиска. Однако для системы Sphinx есть одно отличие, которое существенно снижает гибкость работы – это обязательное наличие схемы данных, которые предполагается хранить и обрабатывать. Данным недостатком можно пренебречь в случае работы со структурированными данными с заведомо известным количеством полей и их предназначением, однако при работе, к примеру, с потоком логов, что применимо в поддержке программных средств, это является минусом.

Таблица 2 - Сравнительные данные движков полнотекстового поиска

Характеристика \ Продукт	Solr	Elastic Stack	Sphinx
1. Скорость индексации	2.75 МБ/с	3.8 МБ/с	4.5 МБ/с
2. Скорость поиска	25 мс	10 мс	7 мс
3. Размер индекса	20 %	20 %	30 %
4. Особенности текстового поиска	1. full-text 2. autocomplete suggestions 3. faceted 4. multifield 5. synonyms 6. fuzzy 7. highlighting 8. geospatial 9. spell checker	1. full-text 2. autocomplete suggestions 3. faceted 4. multifield 5. synonyms 6. fuzzy 7. geospatial	1. full-text 2. autocomplete suggestions 3. faceted 4. multifield 5. synonyms (called wordforms) 6. geospatial 7. highlighting (called snippets) 8. spell checker (called qsuggest)
5. Индексация в реальном времени	Да	Да	Да
6. Поддерживаемые операционные системы	Любая, с JVM	Любая, с JVM	FreeBSD, Linux, NetBSD, OS X, Solaris, Windows
7. Схема данных	Может быть опущена	Может быть опущена	Да
8. Визуализация данных OOB	Нет	Да	Нет
9. Поддерживаемые API	Java API, RESTful HTTP API	Java API, RESTful HTTP/JSON API	Запатентованный протокол
10. Поддержка скриптов на стороне хранилища	Java плагины	Да	Нет
11. Язык разработки	Java	Java	C++

Для использования в качестве движка аналитического инструмента сопровождения автор остановил свой выбор на Elasticsearch. Выбор данного продукта обусловлен возможностями продукта «из коробки», такими как:

- графический интерфейс с инструментами построения визуализаций различных уровней сложности;
- быстрая и гибкая настройка реплицируемых данных с различных источников с использованием компонента Logstash без разработки дополнительных сервисов.

При скорости ответа движка Elasticsearch равной 10 мс, учитывая необходимость вывода информации в течение 2 секунд, грубый расчет максимального объема данных при условии прямо-пропорциональной связи времени ответа и объема информации в хранилище, приемлемого для работы при прочих равнозначных условиях, будет следующий:

1. Запас прочности: 20000мс / 10 мс = 2000.
2. Максимальный объем данных: 2000 х 330 мб = 660000 мб ≈ 660 гб.

Такой объем для небольшого программного решения можно назвать достойным: избыточным для хранения бизнес-данных и приемлимым для хранения логирования с сохранением ротации нефильтрованных логов в 4 недели как минимум.

2.2 Моделирование тестовой среды

В исследовательской работе объект моделирования – процесс работы с аналитическим инструментом поддержки программного обеспечения, используемым в информационной системе вымышленного предприятия по работе с интернет-заказами. Моделирование процесса работы с аналитическим инструментом позволит оценить его реализацию с точки зрения применимости в сфере поддержки. Логическая модель системы представлена на рисунке 5.

Рисунок 5 – Логическая модель имитации аналитического процесса работы вымышленного предприятия по работе с заказами

Приложение по работе с заказами для имитирования деятельности условного предприятия выполняет роль генерации исходных данных, используемых для статистического анализа. Генерация данных осуществляется случайным образом, для хранения данных автором выбрана база данных PostgreSQL.

База данных предприятия имеет табличную структуру, минимально необходимую для работы с заказами.

Следующие сущности входят в базу данных:

1. Заказы.
2. Резервации.
3. Клиенты.
4. Адреса.
5. Ошибки.
6. Складские запасы.

А также вспомогательные сущности, необходимые для полного цикла работы с заказами:

1. Резервации.

2. Статусы заказов.
3. Почтовые адреса клиентов.
4. Виды продуктов.

Структура бизнес-данных вымышленного предприятия показана на рисунке 6.

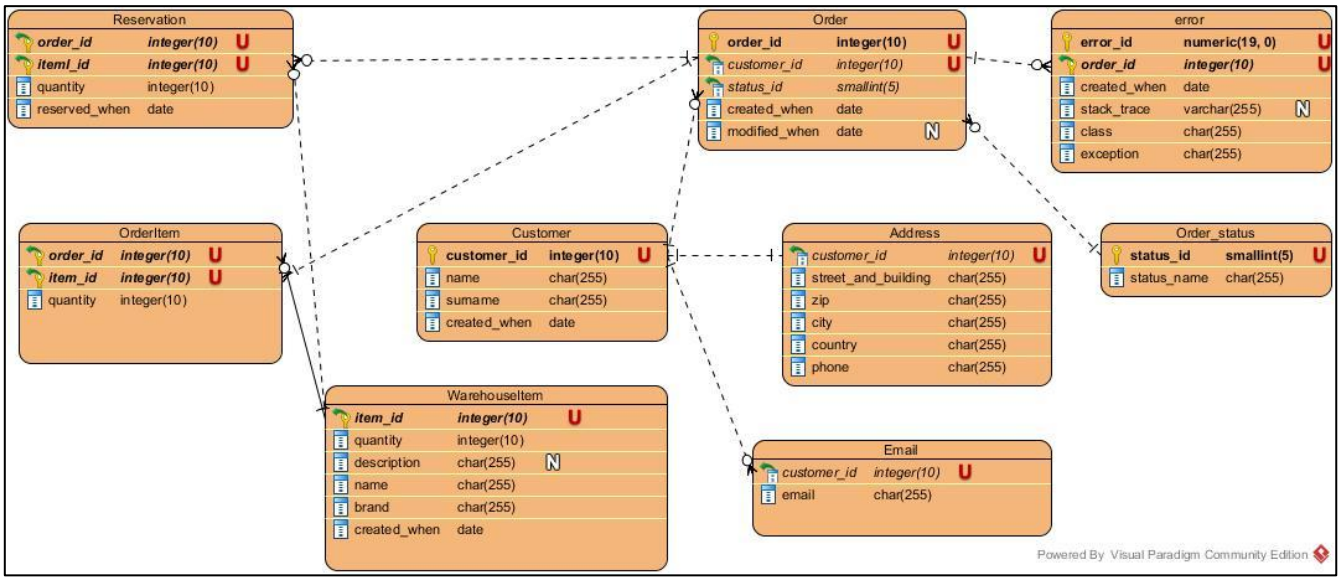

Рисунок 6 - Структура бизнес-данных вымышленного предприятия с полями, ограничениями типов и отношениями между ними

Для генерации тестовых данных автором подготовлена утилита на языке Java, заполняющая таблицы записями. Для генерации дат, количеств использованы псевдослучайные числа. В качестве клиентской базы использовался CSV файл с несуществующими записями клиентов, который предоставляется в бесплатном доступе ресурсом www.briandunning.com. В качестве базы продаваемой продукции в таблицу складских запасов были внесены 18 случайных моделей ноутбуков и периферийных устройств с рынка розничной продажи. Код утилиты представлен в приложении А. На рисунке 7 представлен фрагмент кода утилиты – главного метода управления последовательностью генерации бизнес-сущностей.

Также для проведения исследования были взяты тестовые файлы логирования доступа по протоколу HTTPS для реализации парсинга текстовых файлов и тестирования возможности построения отчетов на основе полученных данных.

```
37      @Override
38      public void run()
39      {
40          for (int i = 0; i < ITEMS_TO_GENERATE; i++)
41          {
42              System.out.println("Creating a Customer");
43              Customer customer = Customer.nextSampleCustomer(client);
44
45              if (customer != null)
46              {
47                  try {
48                      System.out.println("Inserting a customer");
49                      int insertResult = 0;
50                      insertResult = client.insertCustomer(customer);
51                      System.out.println(insertResult + " rows are affected");
52                      System.out.println("Inserting an address");
53                      insertResult = client.insertAddress(customer.address);
54                      System.out.println(insertResult + " rows are affected");
55                      System.out.println("Inserting an email");
56                      insertResult = client.insertEmail(customer.email);
57                      System.out.println(insertResult + " rows are affected");
58                      System.out.println("Inserting customer orders");
59                      client.insertOrders(customer.orders);
60                      System.out.println(insertResult + " rows are affected");
61
62                  } catch (SQLException e) {
63                      e.printStackTrace();
64                  }
65              }
66          }
67      }
```

Рисунок 7 - Фрагмент кода утилиты генерации тестовых данных – главного метода управления последовательностью генерации бизнес-сущностей

Для тестирования возможности построения расчета метрик на базе аналитического инструмента автор выбрал метод расчета показателей KPI [14]. В настоящее время понятие KPI широко распространено и используется во всех технологических сферах. KPI позволяет замерять, оценивать, делать необходимые корректировки и с течением времени вновь замерять, оценивать. Непрерывный

процесс позволяет делать наиболее точные суждения о состоянии продукта или бизнеса и на основе оценки делать выводы о его будущем.

С учетом тестовых данных разработаны показатели KPI работы вымышленного предприятия. Формулы расчета представлены в таблице 3.

Таблица 3 - Формулы расчета показателей KPI для тестовой конфигурации

KPI	Формула расчета (за отчетный период)
1. Процент выполненных заказов >= 90%	**A** – общее количество заказов **B** – выполненные заказы **KPI_1 = B / A * 100**
2. Выполненные заказы в течение 5 дней >= 95%	**C** – количество выполненных заказов **D** – количество выполненных заказов в течение 5 дней **KPI_2 = D / C * 100**
3. Количество проблемных заказов < 5%	**A** – общее количество заказов **E** – проблемные заказы **KPI_3 = E / A * 100**
4. Среднее количество заказов, создаваемых ежедневно >= 10	**A** – общее количество заказов **F** – количество дней **KPI_4 = A / F * 100**

С использованием структуры бизнес-данных подготовлены макеты отчетов для конфигурирования. Отчеты предполагается реализовать в двух выходных форматах: в виде электронного письма и в виде локально сохраненного файла формата CSV. Макеты электронных писем представлены на рисунке 8.

Также для визуализации статистической информации на основе бизнес данных и выполнения функционального требования предполагается реализовать

диаграммы, показанные на рисунке 9. Диаграммы показывают тренды эксплуатации вымышленного программного средства для формирования представления о работе основного бизнес-процесса. Данные построения графиков выбраны в соответствии с сущностями в расчете метрик KPI: заказы, проблемы.

Добрый день,

отчет показателей по заказам за период: ГГГГ-ММ-ДД - ГГГГ-ММ-ДД.

Наименование показателя	Цель	Значение	Отклонение от цели
1. Количество созданных заказов			
2. Процент выполненных заказов	90%		
3. Процент выполненных заказов в течение 5 дней	95%		
4. Количество проблемных заказов			
5. Процент проблемных заказов	5%		
6. Среднее количество заказов, создаваемых ежедневно	10		

Добрый день,

список ошибок выполнения заказов за период: ГГГГ-ММ-ДД - ГГГГ-ММ-ДД.

Класс ошибки	Количество ошибок	Количество проблемных заказов
Класс 1	4	2
Класс 2	6	5

Рисунок 8 - Макет электронного письма с отчетом по KPI показателям и ошибкам работы программного средства

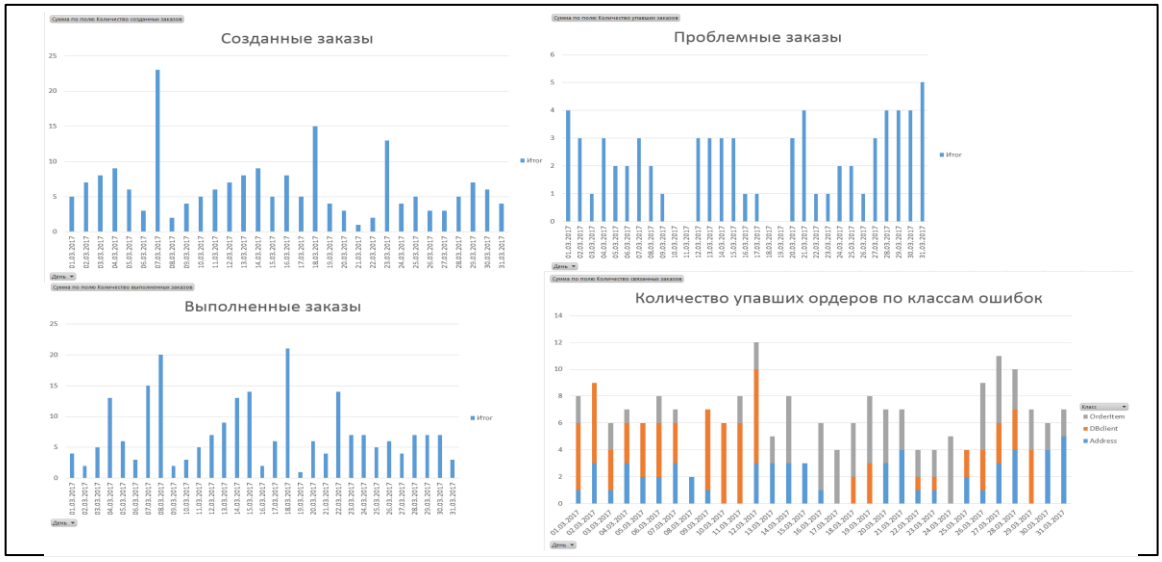

Рисунок 9 - Диаграммы статистических показателей бизнес-данных программного средства

Для подготовки решения исследовательской работы на основе полученных требований выделены следующие задачи:

1. Спроектировать аналитический инструмент на основе движка полнотекстового поиска.
2. Смоделировать тестовую среду для реализации решения в модели процесса поддержки программного средства.
3. Произвести имплементацию модели аналитического инструмента.

Проведенное сравнение трех современных популярных поисковых систем позволяет сделать вывод о том, что рынок программного обеспечения насыщен в полной мере предложениями систем полнотекстового поиска, в том числе и продуктами с открытым исходным кодом. Анализ результатов сравнения показывает наличие достоинств у каждого из рассмотренных продуктов, что говорит о возможности выбора системы под специфические нужды, с учетом технических и функциональных требований. Проектные команды могут свободно выполнять установку, тестирование, доработку без влияния коммерческой выгоды на результаты исследования.

Подготовка тестовой среды для построения отчетов и работы с данными, постановка показателей и методов их расчета позволяет проследить прямую связь между ключевыми бизнес-сущностями программного средства (заказы) и расчетных значений для оценки работы программного решения.

3 ПРЕДСТАВЛЕНИЕ АВТОРСКОГО РЕШЕНИЯ ПОСТАВЛЕННОЙ В ИССЛЕДОВАНИИ ПРОБЛЕМЫ

3.1 Проектирование аналитического инструмента

Выполнение обозначенных требований к аналитическому инструменту реализуется следующими компонентами:

1. Elasticsearch – распределенный полнотекстовой движок с возможностью горизонтального масштабирования, объекты хранятся в виде документов с JSON структурой.
2. Kibana – пользовательский интерфейс для визуального оформления аналитических данных, интегрирующийся с движком Elasticsearch с инструментами конфигурирования и управления.
3. Logstash – динамическая платформа для конфигурирования входных данных на основе подключаемых плагинов, интегрирующаяся с Elasticsearch.
4. FileBeat – сервис для чтения файлов логирования, интегрируется с Logstash.
5. Report Tool – утилита для сборки CSV/email отчетов на основе данных Elasticsearch, работающая по API.
6. Indices Join Tool – утилита для создания временных индексов на основе объединения двух существующих по одному ключевому полю.
7. Cron – демон UNIX-подобных операционных систем для исполнения задач по запланированному графику.

Elasticsearch движок предполагается использовать с однонодовом режиме без настроек на репликацию данных, хотя последняя является рекомендуемым способом реализации отказоустойчивости работы движка вместо использования

RAID-массивов со знаком отличным от нуля [36]. Модель аналитического инструмента показана на рисунке 10.

Конфигурация Java Virtual Machine для компонентов Elastic Stack аналитического инструмента выбрана исходя из минимально необходимых объемов оперативной памяти для работы системы.

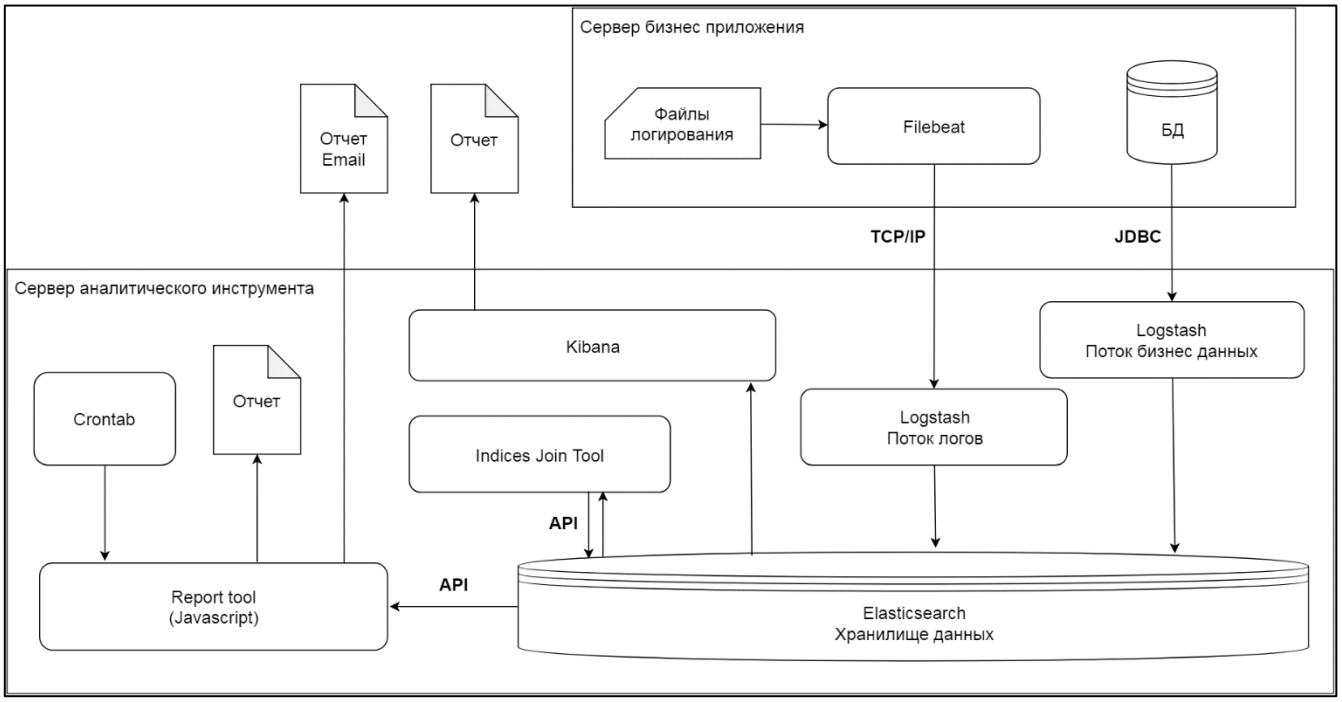

Рисунок 10 - Модель аналитического инструмента в интеграции с бизнес приложением

Ниже приведены значения для стартовой и максимальной памяти Heap по компонентам:

- Elasticsearch (1 нода) – Xms2g, Xmx2g;
- Logstash Logs – Xms512m, Xmx512m;
- Logstash Business Data - Xms512m, Xmx512m.

Утилита Indices Join Tool предназначена для создания временных индексов на основе двух уже существующих. Движок, Elasticsearch, несмотря на многочисленные достоинства по работе с данными методами полнотекстового

поиска, не позволяет объединять индексы данных по соответствиям полей, что является одной из отличительных особенностей NoSQL технологий. Для обхода данной особенности автором предлагается создание утилиты, позволяющей объединять необходимые данные во временные индексы и строить отчеты уже на их основе. Алгоритм работы утилиты представлен на рисунке 11.

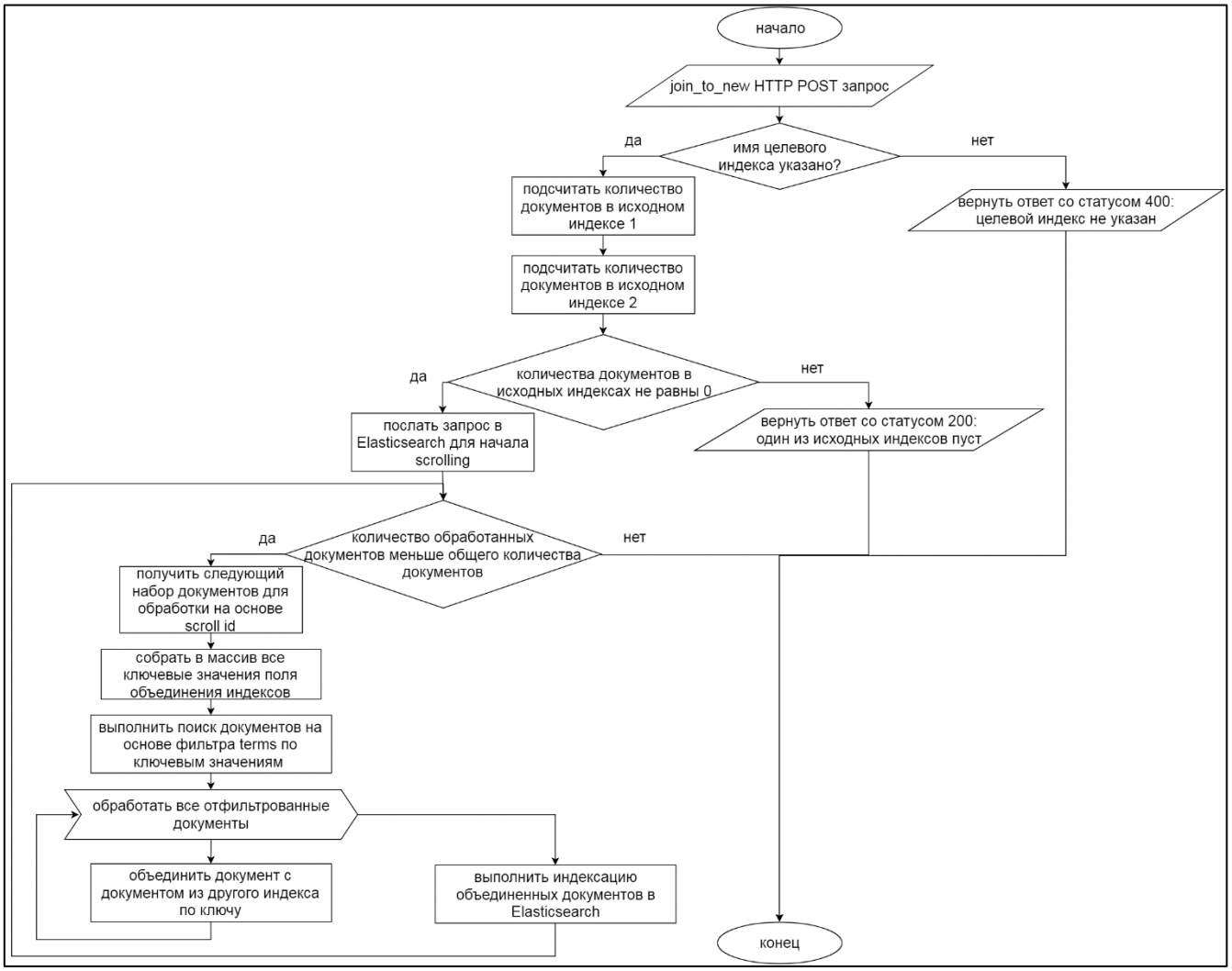

Рисунок 11 – Алгоритм работы утилиты Indices Join Tool

Утилита Report Tool предназначена для сборки готовых отчетов: в виде файлов CSV формата либо электронного сообщения с форматированием в виде HTML разметки. Схема работы утилиты показана на рисунке 12.

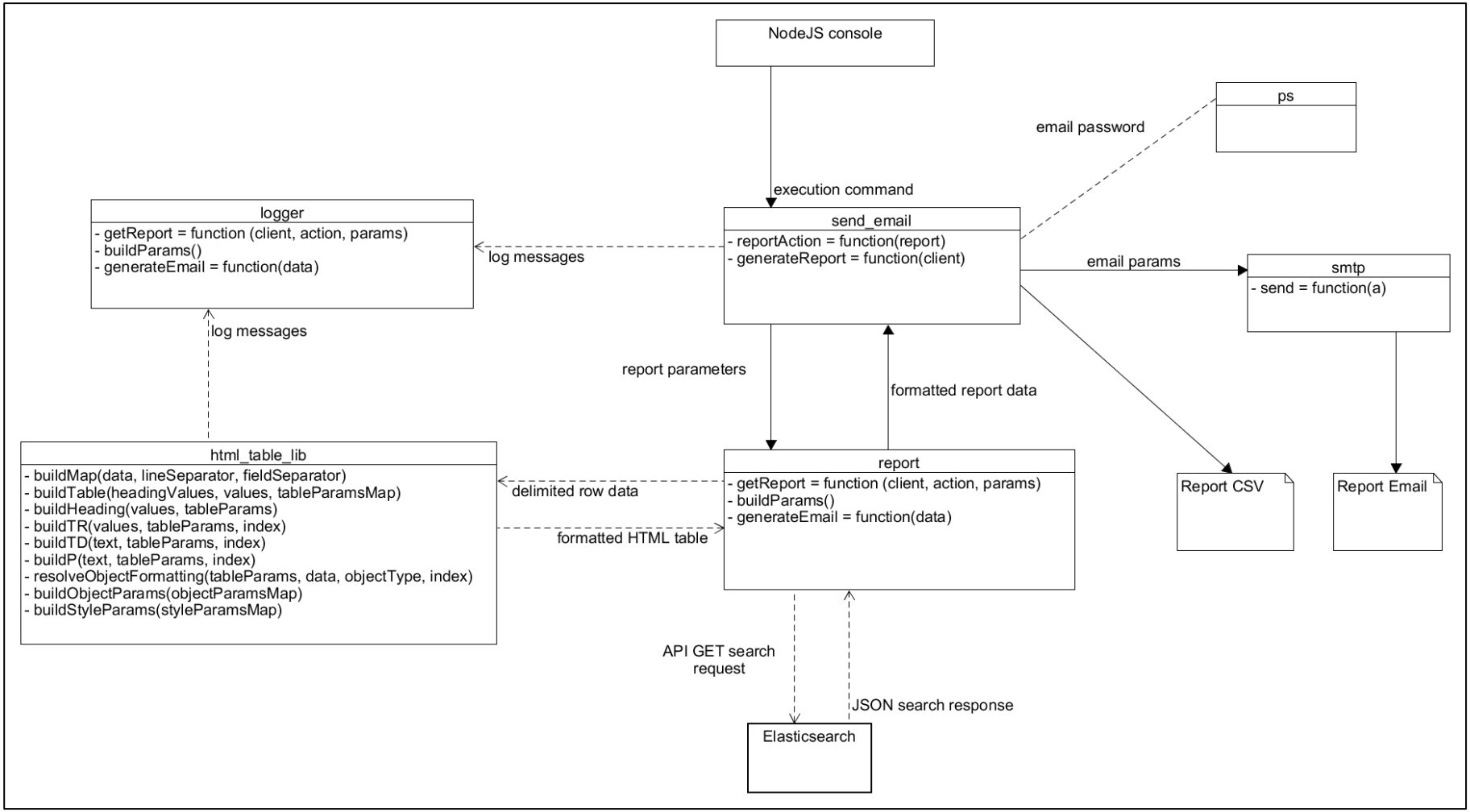

Рисунок 12 - Диаграмма работы утилиты Report Tool – компонента, используемого для сборки отчетов в CSV/email форматах

Report Tool напрямую обращается к Elasticsearch через API библиотеки Javascript посредством посылки запросов на ноду движка. Большинство калькуляций и агрегаций реализуются на стороне системы полнотекстового поиска путем включения в запрос вложенных объектов «aggs», с уровнями вложенности и обязательными параметрами в зависимости от типа агрегации.

В качестве универсального инструмента преобразования полученных данных с разделителями полей и строк реализуется библиотека html_table_lib, упрощающая работу с формированием табличных данных из массивов. Алгоритм работы html_table_lib представлен на рисунке 13. Метод buildMap возвращает массив массивов, созданный из текстовой строки с разделителями полей и строк, что представляет собой табличную структуру данных. Полученный массив можно дополнить, преобразовать, отсортировать, используя стандартные функции работы с array объектами. Дальше готовый к оформлению массив используется в виде параметра при вызове основного метода библиотеки buildTable. Метод возвращает строку с готовой разметкой HTML и позволяет также параметризировать формат отдельных строк либо столбцов, ячеек, путем изменения условного форматирования, который также может указываться в качестве параметра при вызове.

Для своевременной очистки места в хранилище данных автор предусмотрел реализацию shell скриптов с обращением к Elasticsearch посредством API и чисткой необходимых данных с фильтрацией по времени создания документа. Планирование задач чистки также предусмотрено с использованием демона Cron.

Демон Cron используется в качестве планировщика задач на уровне операционной системы. Выполнение задач по сборке отчетов конфигурируется путем написания shell скрипта запуска сборки отчета и конфигурирования запуска данного скрипта с интервалами времени, описанными в cron-формате «* * * * *». Для самопроверки использованного формата использовался открытый ресурс тестирования cron-конфигураций [37].

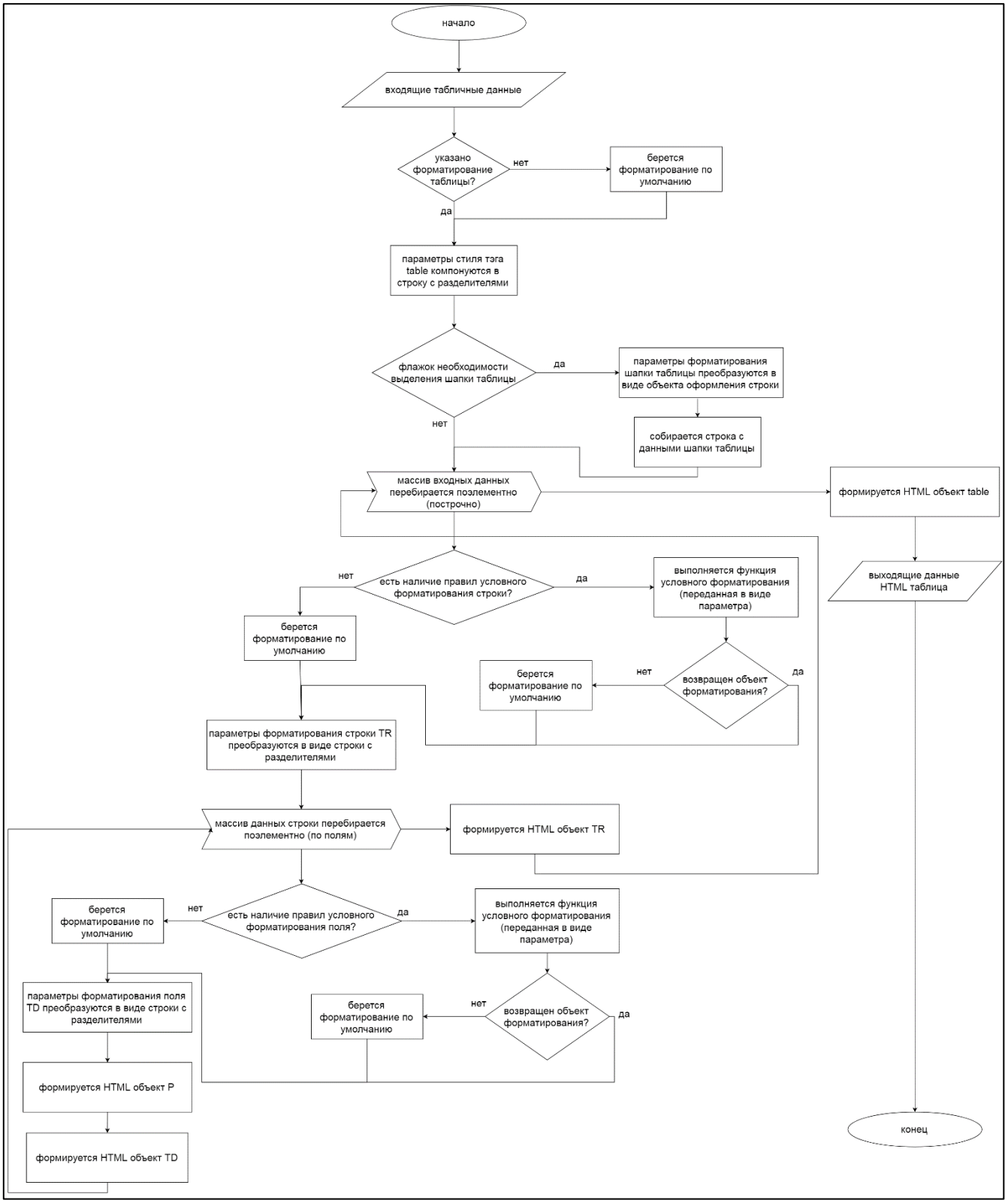

Рисунок 13 - Алгоритм работы сборки форматированной таблицы Javascript библиотеки html_table_lib

На этапе проектирования автором проделана следующая работа:

1. Построена модель аналитического инструмента и его компонентов.
2. Описаны компоненты и способы их реализации.

3.2 Реализация аналитического инструмента

В качестве тестовой платформы автором использована операционная система Micrisoft Windows x64 Home Extended Service Pack 1.

Процесс установки поделен на следующие этапы:

1. Установка и настройка Elasticsearch 5.6.3.
2. Установка и настройка Kibana 5.6.3.
3. Установка и настройка Logstash 5.6.3.
4. Установка и настройка Filebeat 5.6.16.
5. Установка Node.js 8.9.4.
6. Установка и настройка Cron.

В качестве домашней директории аналитического инструмента использован новый каталог в корне одного из дисков файловой системы, обозначаемый далее как $HOME_DIR:

D:\analytic_tool

Также на локальной машине для разработки и запуска утилиты Report tool установлена NodeJS.

3.2.1 Установка и настройка Elasticsearch 5.6.3

Компонент доступен для загрузки по следующей ссылке: https://www.elastic.co/downloads/elasticsearch.

На выходе получен архив, который распаковывается в домашнюю директорию аналитического инструмента. После распаковки необходимо настроить конфигурацию компонента в следующих файлах:

1. $HOME_DIR\elasticsearch-5.6.3\config\jvm.options:

Изменить стартовый и максимальный объем выделяемой оперативной памяти в соответствии с характеристиками локальной машины (выделить 2 Гб):

-Xms2g

-Xmx2g

2. $HOME_DIR\elasticsearch-5.6.3\config\elasticsearch.yml:

   Изменить стандартный порт для обращения к Elasticsearch:

   http.port: 9210

Для запуска используется следующий исполняемый файл, который необходимо выполнить в каталоге $HOME_DIR\elasticsearch-5.6.3\bin:

elasticsearch.bat

После запуска проверить работу можно обращением к Elasticsearch в любом браузере, в ответе на который будет получена информация о кластере:

http://localhost:9210/

3.2.2 Установка и настройка Kibana 5.6.3

Компонент доступен для загрузки по следующей ссылке: https://www.elastic.co/downloads/kibana.

На выходе получен архив, который необходимо распаковать в домашнюю директорию аналитического инструмента. После распаковки необходимо настроить конфигурацию компонента в файле $HOME_DIR\kibana-5.6.3-windows-x86\config\kibana.yml:

1. Изменить стандартный порт для обращения к Kibana:

   server.port: 5610

2. Изменить порт доступа к Elasticsearch:

   elasticsearch.url: "http://localhost:9210"

Для запуска используется следующий исполняемый файл, который необходимо выполнить в каталоге $HOME_DIR\kibana-5.6.3-windows-x86\bin:

kibana.bat

После запуска проверить работу можно обращением к Kibana в любом браузере, в ответе на который будет открыт пользовательский интерфейс:

http://localhost:5610/

3.2.3 Установка и настройка Logstash 5.6.3

Компонент доступен для загрузки по следующей ссылке: https://www.elastic.co/downloads/logstash.

На выходе получен архив, который необходимо распаковать в домашнюю директорию аналитического инструмента. После распаковки необходимо настроить конфигурацию компонента: в файле $HOME_DIR\logstash-5.6.3\config\logstash.yml изменить порт доступа к Elasticsearch:

hosts: ["localhost:9210"]

Для настройки потоков входных данных в Logstash используются плагины входа. С учетом использования базы данных вымышленного предприятия потребуется установка плагина работы с базами данных. Для этой цели будет использован JDBC input plugin. В качестве плагина исходящих данных будет использован предустановленный elasticsearch, который предназначен для отправки обработанных данных в Elasticsearch.

Установка плагина выполняется в каталоге $HOME_DIR\logstash-5.6.3\bin следующей командой:

logstash-plugin install logstash-input-jdbc

Также аккумуляция логов операционной системы с помощью компонента Filebeat потребует Beats input plugin. Данный плагин предустановлен и дополнительных действий не требует.

Для запуска используется следующий исполняемый файл, который необходимо выполнить в каталоге $HOME_DIR\logstash-5.6.3\bin:

logstash.bat –f {conf}

Где {conf} – путь к конфигурационным файлам потока Logstash.

Согласно модели автором предложено разделить потоки данных на два инстанса Logstash: для бизнес-данных и для логов. Такой подход позволит избежать проблем с изменениями конфигурации и переполнением данных: в случае необходимости отключить, к примеру, затягивание логов в аналитический инструмент будет достаточно остановить поток Logstash, ответственный за принятие данных логирования.

Обработка данных в Logstash конфигурируется в текстовых файлах, изменения файлов отслеживаются инстансом, поэтому изменения подхватываются без перезагрузки компонента. Структура файла представляет собой описания в виде конфигурационных объектов последовательности операций: вход данных, обработка фильтрами, выход данных. В качестве входа и выхода могут использоваться различные системы. В аналитическом инструменте используются два типа входов: JDBC и beats, работающие с базой данных и с filebeat соответственно. Библиотека JDBC для Postgres Database 9.6 загружена с официального сайта разработчика https://jdbc.postgresql.org/.

Пример SQL скрипта для репликации заказов показан на рисунке 14, конфигурации репликации заказов на рисунке 15.

```
1  select orders.*
2  , order_status.status_name status_name
3  , case order_status.status_name
4      when 'Delivered' then orders.modified_when
5    else null
6    end completed_when
7  from orders
8  , order_status
9  where modified_when > :sql_last_value
10 and orders.status_id = order_status.status_id
11 order by modified_when
12
```

Рисунок 14 – SQL скрипт для репликации заказов в Elasticsearch, файл «order.sql»

```
input {
  jdbc {
    jdbc_driver_library =>
    "D:\logstash-5.5.2\input_configs\postgresql-42.1.4.j
    ar"
    jdbc_driver_class => "org.postgresql.Driver"
    jdbc_connection_string =>
    "jdbc:postgresql://localhost:5432/kis_task6"
    jdbc_user => "postgres"
    jdbc_password => " postgres"
    schedule => "* * * * *"
    statement_filepath =>
    "D:\analytic_tool\logstash-5.6.3\plugin_sql_scripts\
    customer.sql"
    sql_log_level => "debug"
    tracking_column => "modified_when"
    last_run_metadata_path => "
    D:\logstash-5.5.2\metadata\order"
    id => "customer_in"
    type => "customer"
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9210"]
    index => "customer"
    document_id => "%{customer_id}"
    id => "customer_out"
  }
  stdout { codec => rubydebug }
}
```

Рисунок 15 – Конфигурация Logstash для репликации заказов в Elasticsearch, файл «order-jdbc.conf»

Опция «schedule» описывает частоту опроса базы данных в формате cron. Параметр «tracking_column» позволяет использовать количественные значения полей для фильтрации дельты с момента последней загрузки данных. В SQL запросе в качестве значения для фильтрации берется параметр «:sql_last_value».

Последнее актуальное значение для фильтрации дельты хранится по пути, указанном в параметре «last_run_metadata_path».

Для работы с парсингом строк файлов логирования в Logstash используется фильтр grok, описывающий структуры строки функциями regexp.

Пример фильтра Logstash для разложения строки серверных access логов показан на рисунке 16.

```
filter
{
    if ([@metadata][type] == "access_log")
    {
        grok
        {
            id => "filter_grok_access_log"
            match => { "message" => [
            "^(?<timestamp>\S+\t\S+)\t(?<ip>[^\t]+)\t(?
            <method>[^\t]+)\t(?<status>[^\t]+)\t(?<byte
            s>[^\t]+)\t(?<time_taken>[^\t]+)\t(?<conten
            t_length>[^\t]+)\t(?<user>[^\t]+)\t(?<tid>[
            ^\t]+)\t(?<parent_span>[^\t]+)\t(?<span>[^\
            t]+)\t(?<sampled>[^\t]+)\t(?<X_BSO_USER>[^\
            t]+)\t(?<uri>[^\t]+)\t(?<session>[^\t]+)\t(
            ?<id>[^\t]+)\t(?<tab>[^\t]+)\t(?<action>[^\
            t]+)\t(?<details>[^\t]+)\t(?<request_id>[^\
            t]+)\t(?<operation_id>[^\t]+)\t(?<X_BSO_PRO
            CESS_ID>[^\t]+)\t(?<pageurl>[^\t]+)\t\"?(?<
            referer>[^\t]+(?<!\"))\"?$",
            "%{COMMONAPACHELOG}"] }
            remove_field => [ "message" ]
        }
    }
}
```

Рисунок 16 – фильтр Logstash для обработки строки серверных access логов

Фильтр grok используется для парсинга исходной строки по полям с применением функций regexp, описывающих характер и размер данных.

В приведенном примере описаны следующие поля:

1. timestamp – дата.
2. ip – IP адрес клиента.
3. method – http метод доступа.
4. status – статус команды доступа.
5. bytes – количество переданных байтов.
6. time_taken – затраченное время на выполнение команды доступа.
7. content_length – длина переданного контента.
8. user – имя пользователя.
9. tid – идентификатор пользовательской сессии.
10. parent_span – имя родительского контейнера.
11. span – имя контейнера.
12. sampled – маркер теста.
13. x_bso_user – внутренний пользователь.
14. uri – идентификатор ресурса.
15. session – сессия внутреннего пользователя.
16. id – идентификатор сессии внутреннего пользователя.
17. tab – вкладка контейнера.
18. action – выполняемое действие.
19. details – детали запроса.
20. request_id – идентификатор запроса.
21. operation_id – идентификатор операции.
22. x_bso_process_id - идентификатор процесса.
23. pageurl – ссылка доступа.
24. referer – ссылочный объект.

Пример тестовой исходной строки серверных access логов показан на рисунке 17.

```
284 2019-03-19  08:33:42    10.238.10.46    GET 200 101
    0.002   5743    "administrator" 13dgxuhssier
    orderTools      information false   /CSServlet
    38475762929133229 85    -   -   orders  -   -   UjiOki8
    398743  u0ikJ98 "
    http://10.106.1.69:6300/common/uobject.jsp?tab=_Parameters&object=85375899210133228 51" 85375899210133228 51
```

Рисунок 17 – Тестовая исходная строка серверных access логов

Исходная строка access лога имеет трудночитаемый формат и несет только сухой остаток необходимых данных о действии пользователя. В качестве разделителей полей используется пробел.

3.2.4 Установка и настройка Filebeat 5.6.16

Компонент доступен для загрузки по следующей ссылке: https://www.elastic.co/downloads/past-releases/filebeat-5-6-16.

На выходе получен архив, который необходимо распаковать в домашнюю директорию аналитического инструмента. После распаковки необходимо настроить конфигурацию компонента:

- $HOME_DIR\filebeat-5-6-16-windows-x86_64\filebeat.yml:
  - Задать тип данных, путь к файлам для обработки:

    - input_type: log

      paths:

      - $HOME_DIR/filebeat_test/test_source_data_2/*.csv

      document_type: log_data_event
  - Раскомментировать и заполнить хост и порт доступа к Logstash:

    output.logstash: hosts: ["localhost:5054"]

Для запуска используется следующий исполняемый файл, который необходимо выполнить в каталоге $HOME_DIR\filebeat -5.6.3-windows-x86_64\bin с указанием пути до конфигурационного файла:

```
filebeat.exe -c .\filebeat.yml –e
```

3.2.5 Установка Node.js 8.9.4

Платформа доступна для загрузки по ссылке: https://nodejs.org/en/download.

На выходе получен инсталляционный файл msi, который необходимо запустить для установки платформы Javascript на локальный компьютер. Следуя шагам установки, необходимо задать путь для распаковки файлов, опционально выбрать раздел меню Пуск для создания ярлыков.

Проверка работоспособности Node.js производится выполнением команды вывода версии платформы на экране в командной строке Microsoft Windows:

```
node -v
```

3.2.6 Установка и настройка Cron

Инсталляция демона выполнена автором по рекомендациям в Интернет-ресурсе habr [17]. В качестве задач для исполнения написаны shell скрипты сборки отчетов через утилиту Report Tool.

Пример скрипта сборки отчета по статистике заказов за март 2019 года:

```
node send_email.js report2 true "2019-03-01" "2019-03-31"
```

Команда планирования запуска сборки отчетов каждый понедельник в 9 часов утра 00 минут:

```
0 9 * * 1 \analytic_tool\report_scripts\orders_report.sh
```

Имплементация shell скрипта очистки индексов с использованием API Elasticsearch на примере зачистки ошибок, созданных более чем 7 дней назад, показана на рисунке 18.

```
#!/bin/bash
curl -XPOST "
http://localhost:9201/error/_delete_by_query?refresh&sli
ces=5&pretty" -H 'Content-Type: application/json' -d'
{
    "query": {
        "bool": {
            "must": [
            {
                "range": {
                    "datetime": {
                        "lte": "now-7d"
                    }
                }
            }
            ]
        }
    }
}'
curl -XPOST
"http://localhost:9201/error/_forcemerge?only_expunge_de
letes=true"

```

Рисунок 18 – Shell скрипт очистки ошибок, созданных более чем 7 дней назад

Команда «/_delete_by_query», использованная для очистки старых данных, фактически не выполняет очистку места и перестроение индексов. Для окончательного освобождения места используется последующее выполнение команды «/_forcemerge?only_expunge_deletes=true».

Утилита Report tool написана на языке Javascript согласно разработанной диаграмме и алгоритму работы. Код утилиты представлен в приложении В. Утилита Indices Join Tool также написана на языке Javascript согласно разработанной диаграмме и алгоритму работы. Код утилиты представлен в приложении С.

В графическом интерфейсе Kibana реализованы диаграммы статистических трендов. Их конфигурации представлены на рисунке 19. В зависимости от типа тренда выбраны уровни и поля агрегаций, дополнительные фильтры.

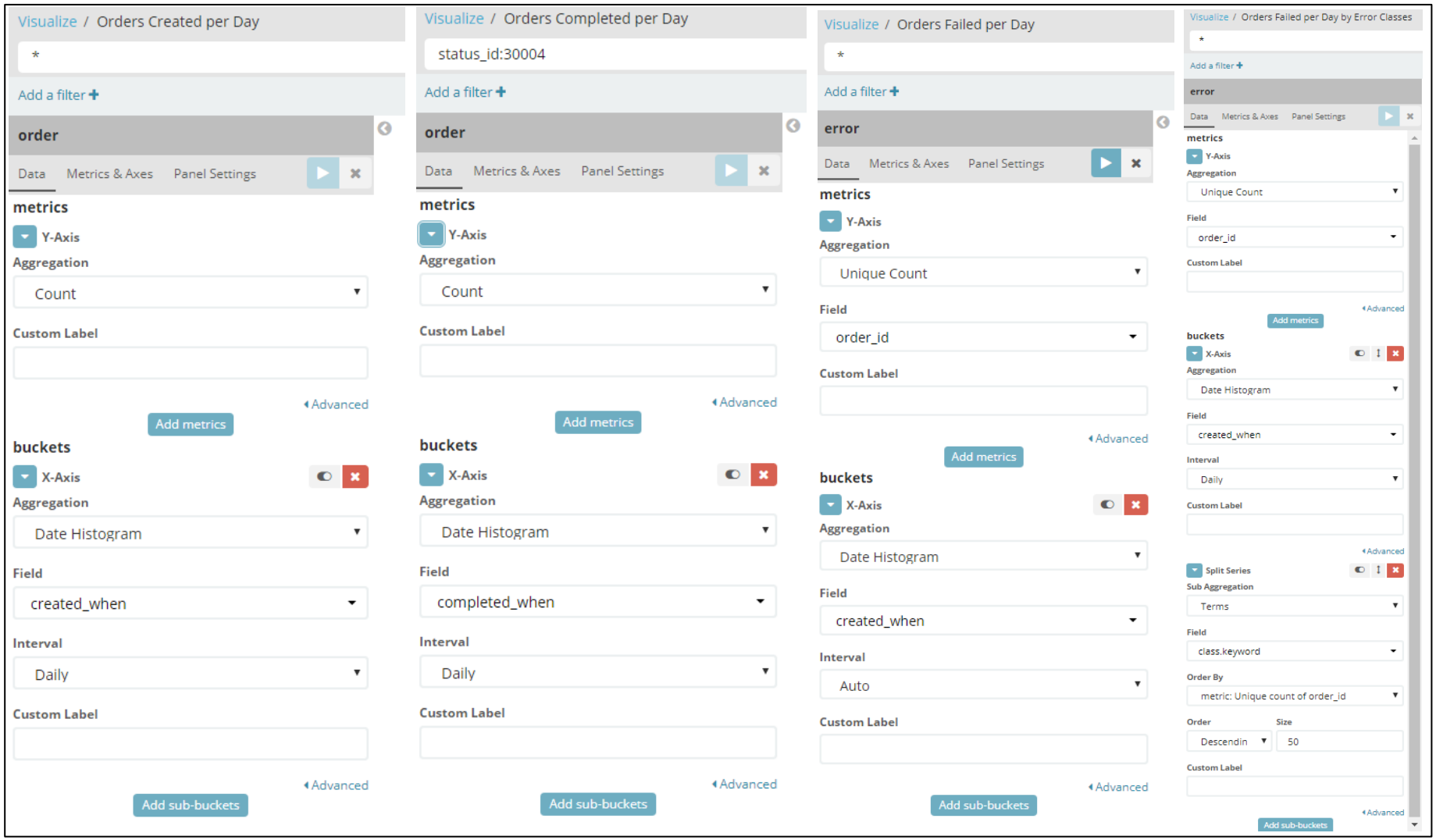

Рисунок 19 - Имплементация диаграмм для построения статистических трендов по бизнес-данным

На данном этапе автором исследования проделана следующая работа:

1. Установлены и сконфигурированы компоненты Elastic Stack.
2. Установлены и сконфигурированы NodeJS, Cron.
3. Имплементирован компонент Report tool с отчетами согласно калькуляционным формулам расчета KPI.
4. Реализованы скрипты для очистки места в хранилище данных, сборки отчетов.
5. Реализованы диаграммы построения трендов по бизнес-данным.

В дальнейшем необходимо:

1. Провести тестирование полученного аналитического инструмента на предмет консистентности данных, соответствия статистики.
2. Проверить соответствие реализованного продукта поставленным функциональным требованиям.
3. Оценить полученное решение с точки зрения применимости в процессах поддержки.

Основываясь на полученном опыте установки и конфигурирования продукта Elastic Stack, автор делает вывод о том, что базовая настройка подобной системы не должна вызывать сложностей и может применяться при необходимости быстрого получения результатов. Организация такого инструмента потребует аппаратных ресурсов, обособленных от ресурсов бизнес-решения. Наличие API для работы с хранилищем данных позволяет выполнять интеграцию движка с уже существующими системами. Разработка новых компонентов с использованием API не вызывает сложностей и имеет широкий спектр работы с поисковым движком.

4 ПРЕДСТАВЛЕНИЕ ЭКСПЕРИМЕНТАЛЬНЫХ И РАСЧЕТНЫХ РЕЗУЛЬТАТОВ АПРОБАЦИИ

4.1 Проверка соответствия решения заданным требованиям

Тестирование аналитического инструмента автор разделил на следующие разделы по объектам проверок:

1. Консистентность данных:
    a. Количество документов (записей).
    b. Заполненность полей (выборочная проверка, проверка всех записей на отсутствие полей).
2. Покрытие функциональных требований:
    a. Работа с большими объемами данных (при максимально возможном объеме, учитывая ограничения используемого персонального компьютера).
    b. Нагрузка на базу данных.
    c. Доступ к информации (поиск по ключу) менее 2 секунд.
    d. Построение визуальных агрегированных отчетов.
    e. Достоверность показателей файлов отчетов, отчетов электронной почты.
    f. Выполнение плановых задач (сборка отчетов, очистка места).
    g. Объединение индексов по ключевому полю.

Далее автором приведены результаты проверок с описанием проведенных тестов.

4.1.1 Количество документов

Проверка количеств документов производилась выполнением скриптовых запросов в базе данных Postgres и в хранилище аналитического инструмента

Elasticsearch. К каждому из типов документов составлены простые запросы подсчета количеств: для базы данных – в таблице, для полнотекстового движка – в соответствующем индексе.

Пример выполненных скриптов по подсчету количеств ошибок с результатами проверки показан на рисунке 20.

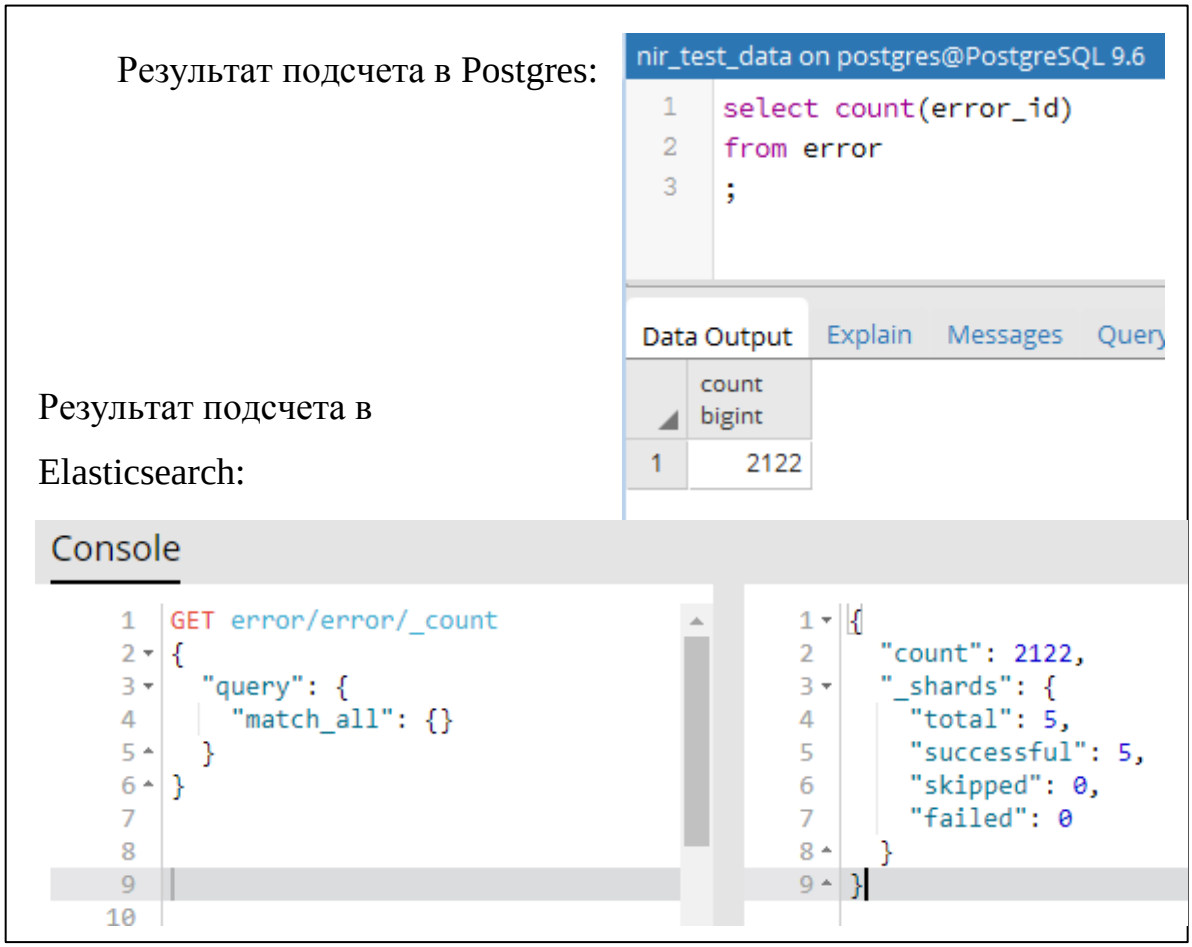

Рисунок 20 - Результаты и использованные скрипты по подсчету количеств ошибок в базе данных и аналитическом инструменте

4.1.2 Заполненность полей

Проверка заполненности реплицированных документов проводилась по всем типам сущностей в индексах Elasticsearch. Тестовый скрипт выполнял подсчет

количества документов с фильтрацией по отсутствующим полям. Список полей предварительно был составлен на основе модели данных. Часть тестового скрипта по проверке полноты данных представлена на рисунке 21.

Рисунок 21 - Тестовый скрипт и результаты проверки отсутствия полей в данных Elasticsearch

4.1.3 Работа с большими объемами данных, доступ к информации (поиск по ключу) менее 2 секунд

Для выполнения замеров времени выполнения запросов поиска по ключевым словам один из индексов был наполнен тестовыми данными до допустимого на тестовом окружении объема памяти в 21.2gb. Замеры вывода данных проводились в браузере Google Chrome версии 72.0.3626.121 (Official Build) (64-bit).

Результаты замеров показаны на рисунке 22, описание выполненных операций:

1. Запрос на открытие индекса без фильтров.
2. Запрос на поиск данных индекса с фильтром по ключевому слову без указания поля.
3. Запрос на поиск данных индекса с фильтром по двум ключевым словам без указания полей.
4. Запрос на поиск данных индекса с фильтром по трем ключевым словам без указания полей.

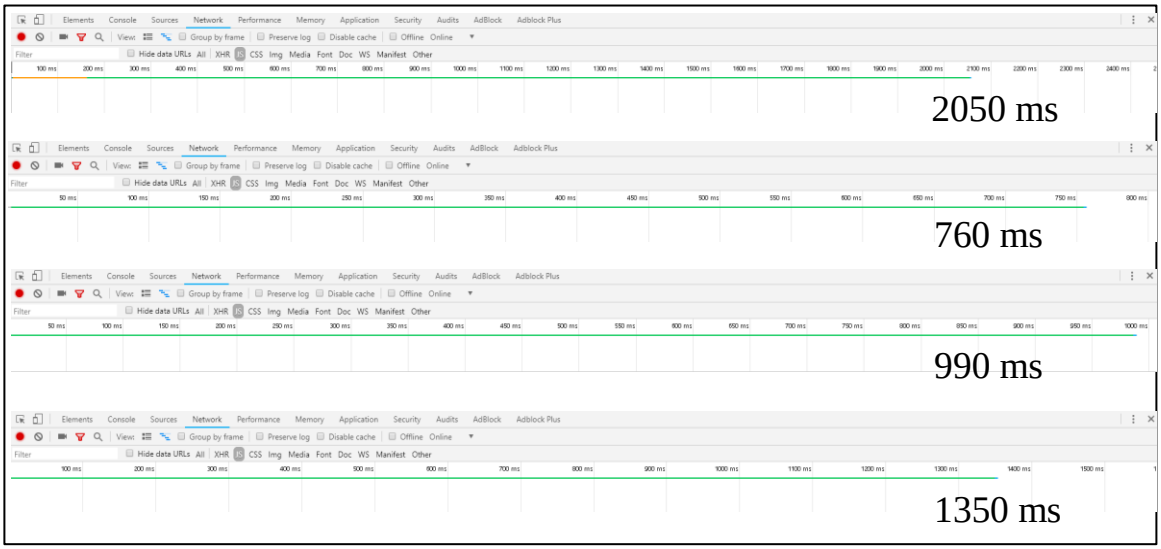

Рисунок 22 - Результаты замеров времени выполнения запросов поиска по ключевым словам из пользовательского интерфейса компонента Kibana

Время отклика показывает, что из протестированных вариантов фильтрации вариантов хуже всего полнотекстовой движок справился с выводом документов без указания фильтров. Это объясняется тем, что индексация, на основе которой построены методы поиска необходимой информации, в данном случае неприменима. Поэтому результат поиска с указанием одного ключевого слова существенно разнится – 760 миллисекунд против 2050. Далее с увеличением количества ключевых слов виден прирост во времени ответа соответственно.

4.1.4 Нагрузка на базу данных

Автору не удалось выполнить полное тестирование нагрузки на базу данных с учетом работающих бизнес процессов и одновременной репликации данных в Elasticsearch ввиду ограниченности возможностей тестового окружения. Для формирования представления о времени выполнения запросов на выгрузку дельты собрана статистика выполнения запросов на затягивание данных с интервалами в одну минуту. Время выполнения запросов показано в таблице 4.

Таблица 4 - Время выполнения запросов репликации данных из базы данных программного средства в аналитический инструмент

п/н	datname	pid	username	application_name	query_start	duration (ms)
1	'nir_test_data'	14144	'postgres'	'PostgreSQL JDBC Driver'	'2019-03-26 18:27:00.045889+00'	34
2	'nir_test_data'	13068	'postgres'	'PostgreSQL JDBC Driver'	'2019-03-26 18:27:00.045889+00'	21
3	'nir_test_data'	11008	'postgres'	'PostgreSQL JDBC Driver'	'2019-03-26 18:27:00.11269+00'	11
6	'nir_test_data'	11012	'postgres'	'PostgreSQL JDBC Driver'	'2019-03-26 18:27:00.271897+00'	44
7	'nir_test_data'	11604	'postgres'	'PostgreSQL JDBC Driver'	'2019-03-26 18:27:00.072489+00'	34

Время выполнения каждого из запросов не отличается высокой нагрузкой на базу данных. Помимо отсутствия бизнес нагрузки на базу данных от приложения, это еще объясняется тем, что запросы на репликацию имеют простую структуру с минимумом фильтрации и без соединения таблиц. Такое применение минимизирует ресурсы, затрачиваемые базой данных. В случае применения сложных запросов и частой репликации для содержания данных в актуальном состоянии автор рекомендует провести несколько тестов с имитацией полной рабочей загрузки.

4.1.5 Построение визуальных агрегированных отчетов, достоверность статистики

В соответствии с требованиями, были построены визуальные отчеты для возможности оценки статистических трендов данных Elasticsearch. Сконфигурированные отчеты показаны на рисунке 23.

Kibana 5.6 позволяет строить визуализации 16 видов. Для конфигурации предусмотрены множественные вложения агрегаций, типы калькуляции, персонализированные калькуляционные фильтры на основе search query синтаксиса, цветовое оформление. Проверка данных отчетов проведена на основе собранных SQL запросов к базе данных с теми же уровнями агрегаций и ограничениями по времени возникновения документов.

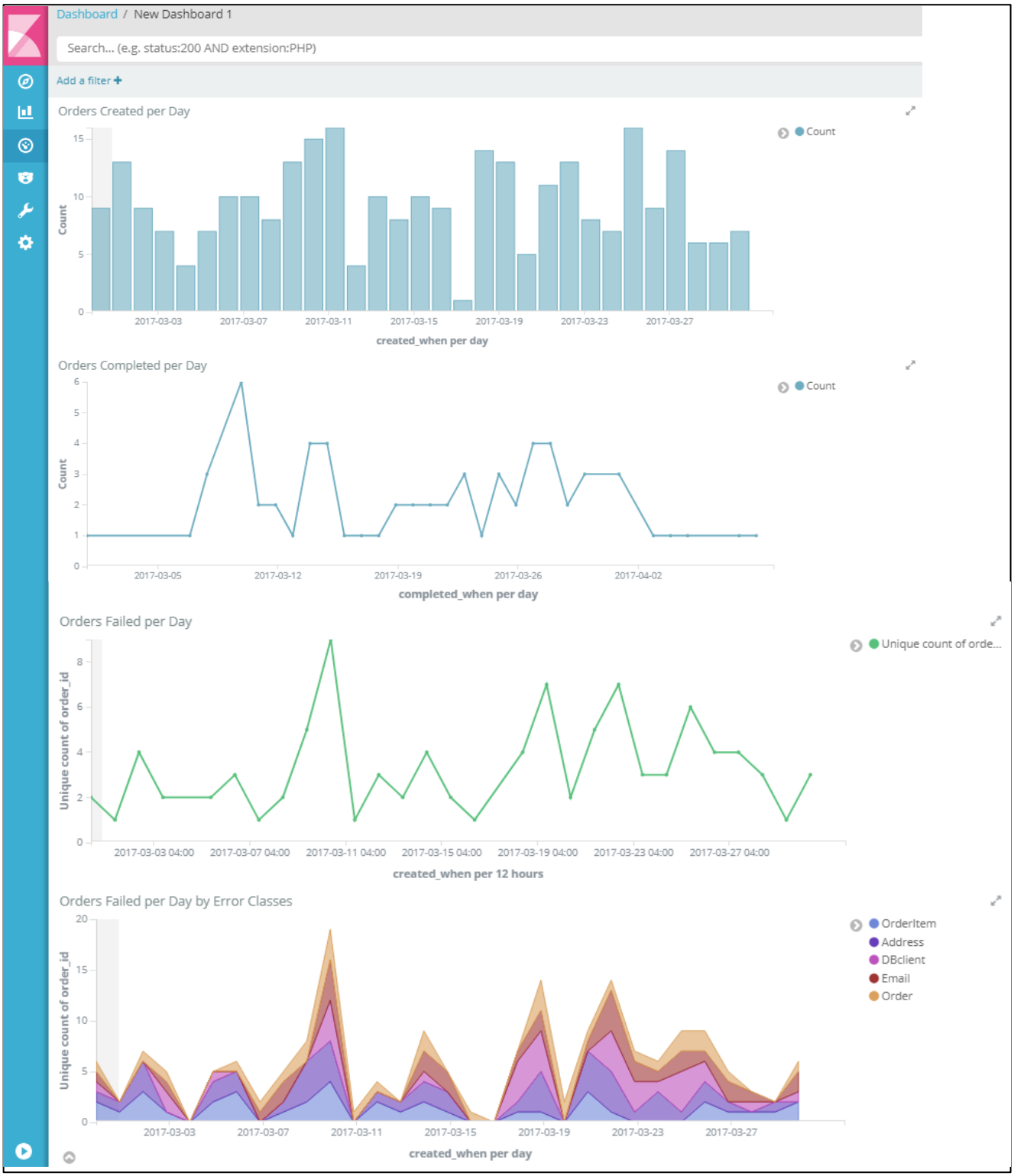

Рисунок 23 – Диаграммы статистических трендов, построенные с использованием компонента Kibana

Пример SQL запроса для проверки количеств созданных заказов, агрегированных по дням создания, показан на рисунке 24.

```
select count(order_id) orders_count
, to_char(created_when, 'YYYY-MM-DD') as date_day
from orders
where created_when >= to_date('2017-03-01', 'YYYY-MM-DD')
and created_when <= to_date('2017-03-31', 'YYYY-MM-DD')
group by to_char(created_when, 'YYYY-MM-DD')
order by to_char(created_when, 'YYYY-MM-DD') asc
;
```

Рисунок 24 – SQL запрос для проверки количеств созданных заказов, агрегированных по дням создания

4.1.6 Выполнение плановых задач (сборка отчетов, очистка места).

Пробные результаты запуска запланированной сборки отчетов с выводом в файл и в виде почтового уведомления показаны на рисунке 25. В ходе тестирования отчетов проблем обнаружено не было. Планировщик задач выполнял команды shell скрипта, который, в свою очередь, вызывал утилиту Report tool с указанием необходимых входных параметров.

Отчет в CSV формате собирался с названиями полей в первой строке и значениями полей, начиная со второй строки, с использованием запятой в качестве разделителя полей. В качестве почтового сервиса для проверки почтовых уведомлений был использован созданный тестовый аккаунт Gmail. В начале письма следует вводная фраза с диапазоном дат, ограничивающих время отчетных данных. Табличные данные оформлены с шапкой наименований столбцов, выделенных жирным шрифтом. Значения полей не содержат искажений данных, порядок строк соблюден.

Отчет формата CSV:

report2.csv

```
1  Класс ошибки,Количество ошибок,Количество проблемных заказов
2  Address,54,45
3  DBclient,49,34
4  OrderItem,44,37
5  Email,40,33
6  Order,33,31
7
```

Отчеты в виде электронных писем:

Отчет Входящие ×

[redacted]@gmail.com

кому: я

Добрый день,

отчет показателей по заказам за период: 2017-03-01 - 2017-03-31

Наименование показателя	Цель	Значение	Отклонение от цели
1. Количество созданных заказов		293	
2. Процент выполненных заказов	0.9	0.23	0.67
3. Процент выполненных заказов в течение 5 дней	0.95	0.60	0.35
4. Количество проблемных заказов		98	
5. Процент проблемных заказов	0.05	0.33	0.28
6. Среднее количество заказов создаваемых ежедневно	10	9.45	0.55

Отчет Входящие ×

[redacted]@gmail.com

кому: я

Добрый день,

список ошибок выполнения заказов за период: 2017-03-01 - 2017-03-31

Класс ошибки	Количество ошибок	Количество проблемных заказов
Address	54	45
DBclient	49	34
OrderItem	44	37
Email	40	33
Order	33	31

Рисунок 25 - Результаты сборки отчетов утилитой Report tool

4.1.7 Объединение индексов по ключевому полю.

Тестирование объединения двух индексов с помощью утилиты Indices Join Tool сделано на основе сгенерированных бизнес-данных: произведено объединение документов индекса error с документами индекса order по условию совпадения значений поля order_id.

Команда выполнения объединения и примеры полученных документов показаны на рисунке 26. В запросах на ограничение документов для обоих объединяемых индексов был использован параметр «match_all», обозначающий использование всех доступных документов в индексе. Из необходимых полей, которые следует перенести в целевой, создаваемый индекс, выбраны: order_id, class, customer_id.

Результаты проведенных тестов показали соответствие имплементации аналитического инструмента поставленным требованиям. Однако, учитывая специфику работы инструмента, при расширении функционала либо масштабировании объемов данных следует включить в тестирование использование полнотекстового движка Elasticsearch с несколькими нодами с уточнением правил аллоцирования индексов по ним, включая внутреннюю настройку репликации данных, если требуется. Это поможет обеспечить отказоустойчивость и распределить нагрузки.

Также результат успешности реализации показал возможность использования следующего функционала, включенного в требования:

- Гарантированность получения актуальной информации;
- Возможности очистки данных по фильтрам;
- API для работы с хранилищем данных для использования внешними системами;
- Поддержка фильтрации входных данных (парсинг текстовых данных в том числе).

Команда инициализации процесса объединения индексов в новый:

POST localhost:3560/join_to_new

Params Authorization Headers (1) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "source":
    [
    {
        "index": "error",
        "join_field":"order_id",
        "query": {
            "match_all": {}
        },
        "fields": ["order_id", "class"]
    },
    {
        "index":"order",
        "join_field_unique":"order_id",
        "query": {"match_all": {}},
        "fields":["order_id", "customer_id"]
    }
    ],
    "target":
    {
        "name": "joined_index_1"
    }
}
```

Результат объединения, документы нового индекса:

```
{
    "took": 5106,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 9290,
        "max_score": 1,
        "hits": [
            {
                "_index": "joined_index_1",
                "_type": "join_error_order201902",
                "_id": "AWguj_G-2CTpFBJILJYp",
                "_score": 1,
                "_source": {
                    "order_id": 30000021,
                    "class": "DBclient",
                    "customer_id": 2000000005,
                    "order_id_joined": 30000021
                }
            },
```

Рисунок 26 - Команда выполнения объединения индексов с использованием Indices Join Tool и примеры полученных документов

В дальнейшем в рамках исследования необходимо выполнить следующие задачи:

1. Оценить полученное решение с точки зрения улучшений и расширения функционала.
2. Оценить недостатки полученного решения.
3. Оценить потраченное время на реализацию.
4. Сделать вывод о возможности применения данного решения в процессе поддержки программных средств.

4.2 Оценка применимости решения в процессе сопровождения

Полученная реализация аналитического инструмента показала возможности для работы с данными, быстрого поиска, построения отчетов. Однако это далеко не весь функционал, который может иметь место для претворения в жизнь, особенно если речь идет о продукте Elastic Stack, который позволяет дополнять существующие компоненты собственными плагинами: для Logstash, для Kibana.

Автор рекомендует следующие идеи по улучшению аналитического инструмента, которые могут оказаться полезными в сопровождении программных средств:

1. Снятие результатов выполнения агрегированных отчетов в виде слепков статистических данных для сохранения статистической информации по трендам после зачистки исходных данных.
2. Создание сервиса по обработке входящих данных и/или обновлении существующих на основе новых данных (к примеру, расчет времени сессии пользователя или времени выполнения заказа со статуса «Готов к доставке» до «Доставлен»).
3. Добавление нотификаций (к примеру, на почту), необходимых для контроля тех или иных данных.

Помимо улучшений, по опыту эксплуатации Elasticsearch, автор указывает на следующие недостатки продукта, которые могут негативно сказаться на решении применять данный продукт (что, возможно, потребует пересмотра выбора продукта или технологии):

1. Несмотря на минимальные требования в 1 Gb оперативной памяти для одной ноды полнотекстового движка, для полноценного решения этого окажется недостаточно, учитывая что каждый компонент сам по себе тоже требует свободных ресурсов. Для возможности работать с большими объемами данных среднего проекта потребуются вложения

в серверную часть и выделение как дискового пространства, так и оперативной памяти. Однако отчасти это компенсируется отсутствием затрат на оплату продукта – он совершенно бесплатен.

2. Запросы на удаление данных, перестройку индексов выполняются продолжительное время и с учетом постоянного чтения/записи в индексы. Удаление – очень дорогостоящая операция и фактически само удаление происходит не сразу: сначала документы помечаются как удаленные, и только потом удаляются все сразу. Возможно, это будет поправлено в новых мажорных версиях полнотекстового движка Elasticsearch.
3. Достижение лимита используемого места на дисковом пространстве окажется серьезной проблемой, учитывая пункт 2. В данном случае наиболее быстрым решением будет удалить полностью один из индексов командой DELETE без указания фильтров. Так как хранилище построено на файловом уровне, удаление через эту команду выполняется за несколько секунд и позволяет избежать проблем с очисткой места. Худшей ситуацией может стать падение ноды ввиду нехватки места, потому что подняться без очистки места она уже не сможет. В данном случае можно прибегнуть к ручному временному переносу файлов данных с ноды на любой свободный ресурс и последующему поднятию ноды для очистки данных. Затронутые shards не будут учтены хранилищем, но в дальнейшем после возврата перенесенных файлов, будут включены обратно в общее хранилище и доступны для работы.
4. Используемые технологии конфигурирования и разработки требуют специальных знаний и опыта, что может затруднить попытки «сходу» получить персонализированное решение. Однако в случае с Elastic Stack, тем не менее, продукт ставится и запускается в чистом виде с

минимальными действиями со стороны пользователя. В то же время технологии визуализации удобны для создания диаграмм, гибкость инструмента действительно стоит внимания.

5. Используемые API сервисы не имеют мер обеспечения безопасности и авторизации. При условии критичности данных либо наличия личных данных заказчика из программного продукта такой инструмент можно использовать только на внутренних ресурсах компании-разработчика.

В таблице 5 приведены затраты времени на реализацию решения и могут использоваться в качестве справочной субъективной информации.

Таблица 5 - Затраты времени на реализацию аналитического инструмента

Задача	Затраты времени
1. Изучение документации	5 часов
2. Проектирование решения	4 часа
3. Выработка стратегии тестирования	1 час
4. Планирование задач	2 часа
5. Разработка, подготовка, наполнение тестовой среды	10 часов
6. Установка и настройка компонентов	2 часа
7. Конфигурирование компонентов и разработка утилит	36 часов
8. Отладка проблем	5 часов
9. Тестирование решения	3 часа
Итого:	68 часов

Учитывая, что автор выполнял разработку в одиночку, время, затраченное на создание решения, значительно сократится. Однако отсутствие опыта работы с системами полнотекстового поиска может существенно его увеличить. В любом случае, в небольших компаниях данный подход будет выгодным при условии

наличия аппаратных ресурсов, так как получаемый функционал универсален и легко расширяем.

Автором выполнена оценка полученного решения с точки зрения ее недостатков, временных затрат на реализацию, возможных вариантов улучшения и расширения функционала.

На основе анализа полученной модели аналитического инструмента автор делает вывод о том, что системы полнотекстового поиска не только справляются со своей непосредственной задачей по быстрому и индексированному поиску, но и, с учетом приложения дополнительных усилий, могут с успехом использоваться в качестве способа получить аналитическую информацию и понять важные аспекты состояния программного средства и решения в целом. Решающую роль в данном случае играет удобство эксплуатации и конфигурирования с минимальными затратами на работу, что увеличивает эффективность работы команды поддержки и минимизирует усилия, требуемые для выполнения нецелевой работы по сбору статистики, сборке отчетов, построения графиков.

ЗАКЛЮЧЕНИЕ

Тенденции современного информационного сообщества ведут к повышению ценности информации, особенно если эта информация максимально структурирована и имеет выжимку статистических данных за период времени. Генерируемые объемы данных уже не позволяют обходиться простыми средствами в виде записей на бумаге, либо ведением таблицы в MS Excel, масштабы проектов все больше приходят к решению эксплуатации собственных систем для ведения своевременной аналитики и решения проблем на основе обдуманных и взвешенных решений, обработанных данных и предсказаниях последствий. Это объясняет популярность разработки сторонних систем, появление NoSQL технологий, распространение продуктов свободной лицензии и открытого кода, который можно изменить.

Анализ процессов поддержки показал, что, несмотря на развитие программных технологий, есть проблемы, с которыми ежедневно сталкивается персонал компаний разработки при сопровождении программного обеспечения. Для повышения эффективности работы команды поддержки использование средств аналитической поддержки существенно облегчит работу по поиску нужной информации и снизит до минимума затраты времени на получение структурированной информации, учитывая что некоторую информацию невозможно получить из консольных приложений. Технологии полнотекстового поиска позволяют быстро получать данные из массива неорганизованных и неструктурированных документов, избегать работ по составлению структур и объявлению полей, которые необходимо добавить в существующую структуру индексов, что упрощает эксплуатацию и повышает практическую значимость инструментов, основанных на этой технологии.

В ходе исследовательской работы удалось получить модель аналитического инструмента. Модель включает в себя основные компоненты для работы с

поиском данных, создания отчетов, анализа данных, обслуживания работы аналитического инструмента. Тестирование инструмента показало пригодность использования для аналитики в процессах поддержки программных средств. Разработанные утилиты Report Tool, Indices Join Tool, работающие на основе API хранилища данных, могут быть использованы в промышленной эксплуатации в интеграции с Elasticsearch.

Проверка соответствия функциональным требованиям показала, что полученная модель аналитического инструмента позволяет решить проблемы процессов поддержки программных средств, а именно:

- обеспечить проектную команду инструментом централизованного доступа к информации для анализа проблем;
- снизить нефункциональную нагрузку на базы данных в сборе статистики;
- сократить процент времени, затрачиваемый сотрудниками на ведение отчетности.

На основе полученной модели можно сделать вывод о том, что использование программного обеспечения с открытым исходным кодом целесообразно и имеет смысл для выделения ресурсов, даже с учетом необходимости доработки до проектных нужд. Полнотекстовые системы поиска могут использоваться не только по прямому назначению поиска информации, но и для статистических нужд, построению аналитики, ведения мониторинга работы систем.

Полученный опыт в разработке и тестировании модели показал, что к требованиям аналитических инструментов, ввиду специфики сферы применения, следует также отнести (помимо общих требований хранилищ Big Data):

1. Поддержка различных источников данных для сбора информации (в том числе базы данных, текстовые файлы).
2. Минимизация нагрузки на базу данных при репликации данных.
3. Возможности очистки данных по фильтрам.

4. API для работы с хранилищем данных для использования внешними системами.
5. Поддержка построения агрегированных отчетов.
6. Визуальное представление данных.
7. Возможность планирования запусков расчета показателей (для системы мониторинга и отчетности) с выходом в виде файлов отчета либо почтового сообщения.

Реализованный аналитический инструмент может быть использован в поддержке программных средств.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Нормативно-правовые акты

1. ГОСТ Р ИСО/МЭК 12207—2010 Процессы жизненного цикла программных средств.— Введ. 2010—11—30.— М.:Изд—во стандартов, 2011.— С. 105.
2. ГОСТ Р ИСО/МЭК 14764—2002. Информационная технология. Сопровождение программных средств.— Введ. 2002—06—25.— М.:Изд—во стандартов, 2002.— С. 32.

Научная и методическая литература

3. Джалота, П. Управление программным проектом на практике. М.: Изд-во «ЛОРИ».- 2005.- 223 с.
4. Липаев, В.В. Программная инженерия. Методологические основы [Текст]: Учеб. / В. В. Липаев; Гос. ун—т — Высшая школа экономики.— М.: ТЕИС, 2006.— 608 с.— 1000 экз.— ISBN 5—7598—0424—3 (в пер.).
5. Липаев, В.В. Стандартизация характеристик и оценивания качества программных средств // Информационные технологии. Приложение.- 2001.- №4.
6. Селезнев, В.П. Современные технологии разработки и сопровождения специального программного обеспечения систем административно-организационного управления: Учеб. пособ.– М.: МГГУ.- 2007.

Электронные ресурсы

7. Любопытная статистика: на что менеджеры тратят свое рабочее время [Электронный ресурс] / Бочкин А. // RUSBASE.- Режим доступа: https://rb.ru/opinion/time-killing.- (Дата обращения: 07.06.2018).
8. 6 причин, почему государству и бизнесу надо разрабатывать проекты с открытым кодом [Электронный ресурс] / Баранова Н. // Теплица социальных технологий.- Режим доступа: https://te-st.ru/2017/10/16/6-reasons-choose-open-source-software.- (Дата обращения: 07.06.2018).

9. Этапы информационного моделирования [Электронный ресурс] / Липунцов Ю.П. // Научно-технические ведомости СПбГПУ. Экономические науки №6(233) 2015.- Режим доступа: https://cyberleninka.ru/article/v/etapy-informatsionnogo-modelirovaniya.- (Дата обращения: 07.06.2018).

10. PMBOK Guide 6. Руководство к своду знаний по управлению проектами [Электронный ресурс] // Бюро проектов.- Режим доступа: http://projectbureau.ru/standard/pmbok6.- (Дата обращения: 07.06.2018).

11. Методы проактивного мониторинга информационных систем [Электронный ресурс] / Дубровин М.Г., Глухих И.Н. // Системный администратор.- Режим доступа: http://samag.ru/archive/article/3599.- (Дата обращения: 07.06.2018).

12. Устройство полнотекстового индекса [Электронный ресурс] // ruhighload.- Режим доступа: https://ruhighload.com/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9+%D0%BF%D0%BE%D0%B8%D1%81%D0%BA.- (Дата обращения: 10.07.2018).

13. Сравнение движков полнотекстового поиска [Электронный ресурс] // Custis.- Режим доступа: http://lib.custis.ru/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BA%D0%BE%D0%B2_%D0%BF%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0.- (Дата обращения: 10.07.2018).

14. Система KPI (Key Performance Indicator): разработка и применение показателей бизнес-процесса. Показатели эффективности [Электронный ресурс] // Business studio.- Режим доступа:

https://www.businessstudio.ru/articles/article/sistema_kpi_key_performance_indicator_razrabotka_i/.- (Дата обращения: 05.06.2018).

15. Имитационное моделирование [Электронный ресурс] // Информационно-образовательный сайт.- Режим доступа:http://info-tehnologii.ru/IMIT_MOD/index.html.- (Дата обращения: 05.06.2018).

16. Леонид brujeo Быстрый полнотекстовый поиск ElasticSearch [Электронный ресурс] / Леонид brujeo // .- Режим доступа:https://habr.com/ru/post/122531.- (Дата обращения: 05.06.2018).

17. Разворачиваем cron в Windows [Электронный ресурс] / deko // habr.- Режим доступа: https://habr.com/ru/post/149545.- (Дата обращения: 10.06.2018).

18. Документация к PostgreSQL 9.6.12 [Электронный ресурс] // postgrespro.- Режим доступа:https://postgrespro.ru/docs/postgresql/9.6/index.- (Дата обращения: 05.06.2018).

19. Аншина, М. Сопровождение программных систем [Электронный ресурс] / М. Аншина // Управляем предприятием № 05.— 2012.— № 16.– Режим доступа: http://upr.ru/upload/iblock/1f7/anshina_lWdI.pdf.—(Дата обращения: 05.04.2017).

20. Бутузов, К. А. Порядок передачи программных средств и организация сопровождения программного обеспечения [Электронный ресурс] / К. А. Бутузов // Известия Петербургского университета путей сообщения.— 2004.— №1. Режим доступа: http://cyberleninka.ru/article/n/poryadok—peredachi—programmnyh—sredstv—i—organizatsiya—soprovozhdeniya—programmnogo—obespecheniya (Дата обращения: 02.04.2017).

21. Дадыкин, А. К. Современная методика организации процесса разработки программ [Электронный ресурс] / А. К. Дадыкин, Ермолаев А. А. // Системный анализ и прикладная информатика.— 2013.— №1—2. Режим доступа: http://cyberleninka.ru/article/n/sovremennaya—metodika—organizatsii—protsessa—razrabotki—programm (Дата обращения: 02.04.2017).

22. Ефимов, Г. Жизненный цикл информационных систем [Электронный ресурс] / Г. Ефимов // Журнал «Сетевой».— 2001.— № 2.– Режим доступа: http://www.setevoi.ru/cgi—bin/text.pl/magazines/2001/2/44.—(Дата обращения: 04.04.2017).

23. Иванова, Н.Ю. Системное и прикладное программное обеспечение [Электронный ресурс]: учебное пособие/ Иванова Н.Ю., Маняхина В.Г.— Электрон. текстовые данные.— М.: Прометей, 2011.— 202 с.— Режим доступа: http://www.iprbookshop.ru/58201.html.— (Дата обращения: 03.04.2017).

24. Кулагин, О. Какие KPI выбрать и почему [Электронный ресурс] / О. Кулагин // Корпоративный менеджмент.- 2014.- Режим доступа: http://www.cfin.ru/management/controlling/kpi_choice.shtml.— (Дата обращения: 04.04.2017).

25. Орлик, С. Программная инженерия. Сопровождение программного обеспечения (Software Maintenance) [Электронный ресурс] / С. Орлик.- 2004-2005.- Режим доступа: http://software-testing.ru/files/se/3-5-software_engineering_maintenance.pdf.- (Дата обращения: 04.04.2017).

26. Системы принятия решений [Электронный ресурс]: учебно-методический комплекс по специальности 080801 «Прикладная информатика (в информационной сфере)», специализации «Информационные сети и системы», квалификация «информатик-аналитик»/ — Электрон. текстовые данные.— Кемерово: Кемеровский государственный институт культуры, 2013.— 56 с.— Режим доступа: http://www.iprbookshop.ru/29703.html.— (Дата обращения: 05.04.2017).

27. Славков, С. А. Анализ систем ведения проектов и багтрекинга для последующей интеграции [Электронный ресурс] / С. А. Славков // Научный журнал Novainfo.ru.— 2016.— №46—4.— Режим доступа: http://novainfo.ru/article/6377.— (Дата обращения: 03.04.2017).

28. Смирнов, А.А. Прикладное программное обеспечение [Электронный ресурс]: учебное пособие/ Смирнов А.А.— Электрон. текстовые данные.— М.: Евразийский открытый институт, 2011.— 384 c.— Режим доступа: http://www.iprbookshop.ru/11079.html.— (Дата обращения: 03.04.2017).

29. Фрэнкс, Б. Укрощение больших данных: как извлекать знания из массивов информации с помощью глубокой аналитики [Электронный ресурс]: / Б. Фрэнкс.— Электрон. текстовые данные.— М.: Манн, Иванов и Фербер, 2014.— 340 c.— Режим доступа: http://www.iprbookshop.ru/39433.html.— (Дата обращения: 03.04.2017).

30. Хан, В. Л. Сопровождение систем автоматизации программного обеспечения [Электронный ресурс] / В. Л. Хан, А. Кизим // Молодой ученый.— 2011.— №5. Т.1.— С. 110—112.— Режим доступа: https://moluch.ru/archive/28/3173.— (Дата обращения: 04.04.2017).

Литература на иностранном языке

31. Number of available applications in the Google Play Store from December 2009 to December 2018 [Электронный ресурс] // Statista.- Режим доступа: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store.- (Дата обращения: 05.06.2018).

32. Google Play Categories [Электронный ресурс] // 42 matters.- Режим доступа: https://42matters.com/docs/app-market-data/android/apps/google-play-categories.- (Дата обращения: 05.06.2018).

33. DB-Engines Ranking of Search Engines [Электронный ресурс] // 14. DB-Engines.- Режим доступа: https://db-engines.com/en/ranking/search+engine.- (Дата обращения: 10.07.2018).

34. Elasticsearch vs. solr vs. sphinx: best open source search platform comparison [Электронный ресурс] / Klimenko A. // greenice.- Режим доступа: https://greenice.net/elasticsearch-vs-solr-vs-sphinx-best-open-source-search-platform-comparison.- (Дата обращения: 10.07.2018).

35. Supporting Your Product: How To Provide Technical [Электронный ресурс] // Smashing magazine.- Режим доступа: https://www.smashingmagazine.com/2011/10/supporting-product-providing-technical-support.- (Дата обращения: 05.06.2018).

36. Elasticsearch: The Definitive Guide [2.x] » Administration, Monitoring, and Deployment » Production Deployment » Hardware [Электронный ресурс] // elasticsearch docs.- Режим доступа: https://www.elastic.co/guide/en/elasticsearch/guide/current/hardware.html.- (Дата обращения: 10.06.2018).

37. The quick and simple editor for cron schedule expressions by Cronitor [Электронный ресурс] // crontab guru.- Режим доступа: https://crontab.guru.- (Дата обращения: 11.06.2018).

38. Apache Solr vs Elasticsearch [Электронный ресурс] // solr-vs-elasticsearch.com.- Режим доступа:http://solr-vs-elasticsearch.com.- (Дата обращения: 05.06.2018).

39. Elasticsearch Reference [Электронный ресурс] // elastic.- Режим доступа:https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.- (Дата обращения: 05.06.2018).

40. Kibana User Guide [Электронный ресурс] // elastic.- Режим доступа:https://www.elastic.co/guide/en/kibana/current/index.html.- (Дата обращения: 05.06.2018).

41. Logstash Reference [Электронный ресурс] // elastic.- Режим доступа:https://www.elastic.co/guide/en/logstash/current/index.html.- (Дата обращения: 05.06.2018).

42. Filebeat Reference [Электронный ресурс] // elastic.- Режим доступа: https://www.elastic.co/guide/en/beats/filebeat/5.6/index.html.- (Дата обращения: 05.06.2018).

43. Duvall, PauM. Continuous Integration Improving Software Qualityand Reducing Risk / PauM. Duvall, Stephen Матиас, Andrew Glover. -USA,«Addison-Wesley».- 2007.- 336 с.

44. Brown, D. Reengineering Customer Support [Электронный ресурс] / D. Brown // AFSM International.- November/December.- 2005, с.61 - 64.- Режим доступа: http://www.supportcenteru.com/images/Article-Reengineering6-DesignNewModel.pdf.— (Дата обращения: 05.04.2017).

45. Viswanathan, B. Understanding The Different Levels of Help Desk Support [Электронный ресурс] / B. Viswanathan // Project-Management.com.- 2016.- Режим доступа: https://project-management.com/understanding-the-different-levels-of-help-desk-support.— (Дата обращения: 04.04.2017).

46. Windley, P. J. Delivering High Availability Services Using a Multi-Tiered Support Model [Электронный ресурс] / P. J. Windley.- 2002.- Режим доступа: https://ru.scribd.com/document/129058297/Tiered-Support-pdf.- (Дата обращения: 04.04.2017).

ПРИЛОЖЕНИЕ А

Код java утилиты заполнения таблиц баз данных

Класс NIR_test_data_generator:

```
import java.sql.*;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.TimeZone;

public class NIR_test_data_generator implements Runnable
{

    static final int ITEMS_TO_GENERATE = 500;
    static final String DB_URL =
"jdbc:postgresql://localhost:5432/nir_test_data";
    static final String DB_USER = "postgres";
    static final String DB_PASS = "postgres";
    static final Date OBJECTS_START_DATE = new GregorianCalendar(2017, 0,
1).getTime();
    static Date current_date;

    DBclient client;

    public static void main (String ... args)
    {
        System.out.println("Creating a generator");
        NIR_test_data_generator generator = new NIR_test_data_generator();
        TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
        NIR_test_data_generator.current_date = new Date();

        generator.run();

    }

    NIR_test_data_generator()
    {
        System.out.println("Creating a DBclient");
        client = DBclient.getClient(DB_URL, DB_USER, DB_PASS);
```

```
        System.out.println("Checking the connection");
        client.checkConnection();
    }

    @Override
    public void run()
    {
        for (int i = 0; i < ITEMS_TO_GENERATE; i++)
        {
            System.out.println("Creating a Customer");
            Customer customer = Customer.nextSampleCustomer(client);

            if (customer != null)
            {
                try {
                    System.out.println("Inserting a customer");
                    int insertResult = 0;
                    insertResult = client.insertCustomer(customer);
                    System.out.println(insertResult + " rows are affected");
                    System.out.println("Inserting an address");
                    insertResult = client.insertAddress(customer.address);
                    System.out.println(insertResult + " rows are affected");
                    System.out.println("Inserting an email");
                    insertResult = client.insertEmail(customer.email);
                    System.out.println(insertResult + " rows are affected");
                    System.out.println("Inserting customer orders");
                    client.insertOrders(customer.orders);
                    System.out.println(insertResult + " rows are affected");

                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Класс DBclient:

```
import java.sql.*;
import java.util.Properties;

class DBclient
{
    Connection connection;

    DBclient (Connection connection)
    {
        this.connection = connection;
    }

    public static DBclient getClient(String url, String user, String pass)
    {
        try {
            Properties props = new Properties();
            props.setProperty("user", user);
            props.setProperty("password", pass);
            return new DBclient(DriverManager.getConnection(url, props));
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    void checkConnection()
    {
        try {
            System.out.println(connection.isValid(1000));
        } catch (SQLException e) {
            System.out.println("DB is unavailable");
            e.printStackTrace();
        }
    }

    int insertEmail(Email email) throws SQLException
    {
        String query = "insert into public.email(customer_id, email,
```

```
created_when) " +
                "VALUES (" +
                email.customer_id + "," +
                addQuotes(email.email) + "," +
                "?);";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new java.sql.Date(email.created_when.getTime()));
        return statement.executeUpdate();
    }

    int insertCustomer(Customer customer) throws SQLException
    {
        String query = "insert into public.customer(customer_id, name,
surname, created_when) " +
                "VALUES (" +
                customer.customer_id + "," +
                addQuotes(customer.name) + "," +
                addQuotes(customer.surname) + "," +
                "?);";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new
java.sql.Date(customer.created_when.getTime()));
        return statement.executeUpdate();
    }

    int insertAddress(Address address) throws SQLException
    {
        String query = "insert into public.address(customer_id,
street_and_building, city, country, zip, phone, created_when) " +
                "VALUES (" +
                address.customer_id + "," +
                addQuotes(address.street_and_building) + "," +
                addQuotes(address.city) + "," +
                addQuotes(address.country) + "," +
                addQuotes(address.zip) + "," +
```

```
                addQuotes(address.phone) + "," +
                "?);";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new
java.sql.Date(address.created_when.getTime()));
        return statement.executeUpdate();
    }

    void insertOrders(Order [] orders) throws SQLException
    {
        for (Order order: orders)
        {
            System.out.println("Inserting order #" + order.order_id);
            int insertResult = insertOrder(order);
            System.out.println(insertResult + " rows are affected");
            System.out.println("Inserting order items");
            insertOrderItems(order.order_items);
            System.out.println(insertResult + " rows are affected");
            System.out.println("Inserting errors");
            insertErrors(order.errors);
        }
    }

    int insertOrder(Order order) throws SQLException
    {
        String query = "insert into public.orders(order_id, customer_id,
status_id, created_when, modified_when) " +
                "VALUES (" +
                order.order_id + "," +
                order.customer_id + "," +
                order.status_id + "," +
                "?," +
                "?);";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new java.sql.Date(order.created_when.getTime()));
```

```
        statement.setDate(2, new
java.sql.Date(order.modified_when.getTime()));
        return statement.executeUpdate();
    }

    void insertOrderItems(OrderItem [] orderItems) throws SQLException
    {
        for (OrderItem orderItem: orderItems)
        {
            System.out.println("Inserting orderItem #" + orderItem.item_id);
            int insertResult = insertOrderItem(orderItem);
            System.out.println(insertResult + " rows are affected");
        }
    }

    int insertOrderItem(OrderItem orderItem) throws SQLException
    {
        String query = "insert into public.order_item(order_id, item_id,
quantity, created_when) " +
                "VALUES (" +
                orderItem.order_id + "," +
                orderItem.item_id + "," +
                orderItem.quantity + "," +
                "?);";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new
java.sql.Date(orderItem.created_when.getTime()));
        return statement.executeUpdate();
    }

    void insertErrors(Error [] errors) throws SQLException
    {
        for (Error error: errors)
        {
            System.out.println("Inserting error #" + error.error_id);
            int insertResult = insertError(error);
            System.out.println(insertResult + " rows are affected");
```

```
        }
    }

    int insertError(Error error) throws SQLException
    {
        String query = "insert into public.error(created_when, order_id,
stack_trace, error_id, class, exception) " +
                "VALUES (" +
                "?," +
                error.order_id + "," +
                addQuotes(error.stack_trace) + "," +
                error.error_id + "," +
                addQuotes(error.class_name) + "," +
                addQuotes(error.exception) +
                ");";
        System.out.println("Query to execute:");
        System.out.println(query);
        PreparedStatement statement = connection.prepareStatement(query);
        statement.setDate(1, new java.sql.Date(error.created_when.getTime()));
        return statement.executeUpdate();
    }

    long selectMaxCustomerId() throws SQLException
    {
        String query = "select MAX(customer_id) from customer;";

        ResultSet result = connection.createStatement().executeQuery(query);
        if (result.next())
        {
            return result.getLong(1);
        }
        return 0;
    }

    long selectMaxOrderId() throws SQLException
    {
        String query = "select MAX(order_id) from orders;";

        ResultSet result = connection.createStatement().executeQuery(query);
```

```
        if (result.next())
        {
            return result.getLong(1);
        }
        return 0;
    }

    long selectMaxItemId() throws SQLException
    {
        String query = "select MAX(item_id) from warehouse_item;";

        ResultSet result = connection.createStatement().executeQuery(query);
        if (result.next())
        {
            return result.getLong(1);
        }
        return 0;
    }

    int selectMaxStatusId() throws SQLException
    {
        String query = "select MAX(status_id) from order_status;";
        ResultSet result = connection.createStatement().executeQuery(query);
        if (result.next())
        {
            return result.getInt(1);
        }
        return 0;
    }

    int selectMaxErrorId() throws SQLException
    {
        String query = "select MAX(error_id) from error;";
        ResultSet result = connection.createStatement().executeQuery(query);
        if (result.next())
        {
            return result.getInt(1);
        }
        return 0;
```

```
    }

    String addQuotes(String s)
    {
        return "'" + s + "'";
    }

    static Exception generateException()
    {
        try {
            throw new SQLException();
        }
        catch (Exception e)

        {
            return e;
        }
    }
}
```

Класс Order:

```
import java.sql.SQLException;
import java.util.Date;

class Order
{
    static final long MAX_DIFF_CREATED_WITH_CUST_CREATED_WHEN_MILLIS =
604800000; // 7 days
    static final long MAX_DIFF_MODIFIED_WITH_CREATED_WHEN_MILLIS = 1209600000;
// 14 days
    static final int MIN_STATUS_ID = 30001;
    static final int MAX_ORDER_ITEMS = 10;
    static final int MIN_ORDER_ITEMS = 1;
    static final int MAX_ERRORS = 6;
    static final int MIN_ERRORS = 0;
    static final long INITIAL_ID = 30000000;

    long order_id;
```

```
    long customer_id;
    long status_id;
    Date created_when;
    Date modified_when;
    OrderItem [] order_items;
    Error [] errors;

    private Order(long order_id,
            long customer_id,
            long status_id,
            Date created_when,
            Date modified_when,
            OrderItem [] order_items,
            Error [] errors
    )
    {
        this.order_id = order_id;
        this.customer_id = customer_id;
        this.status_id = status_id;
        this.created_when = created_when;
        this.modified_when = modified_when;
        this.order_items = order_items;
        this.errors = errors;
    }

    static Order generateOrder(long customer_id, Date customer_created_when,
DBclient client, int idShift)
    {
        System.out.println("Getting a max order id");
        try {
            long maxOrderId = client.selectMaxOrderId();
            if (maxOrderId == 0 )
            {
                maxOrderId = INITIAL_ID;
            }
            System.out.println("Max order id = " + maxOrderId);
            long orderId = maxOrderId + 1 + idShift;
            System.out.println("Randomizing a status");
            int status = randomizeStatus(client);
```

```
            System.out.println("Status = " + status);
            System.out.println("Randomizing a created_when date");

            Date max_creation_date = new Date(customer_created_when.getTime()
+ MAX_DIFF_CREATED_WITH_CUST_CREATED_WHEN_MILLIS);
            if (max_creation_date.getTime() >
NIR_test_data_generator.current_date.getTime())
            {
                max_creation_date = NIR_test_data_generator.current_date;
            }

            Date created_when = Random.genRandomDate(customer_created_when,
max_creation_date);
            System.out.println("Created_when = " + created_when.toString());
            System.out.println("Randomizing a modified_when date");

            Date max_modification_date = new
Date(customer_created_when.getTime() +
MAX_DIFF_MODIFIED_WITH_CREATED_WHEN_MILLIS);
            if (max_modification_date.getTime() >
NIR_test_data_generator.current_date.getTime())
            {
                max_modification_date = NIR_test_data_generator.current_date;
            }

            Date modified_when = Random.genRandomDate(created_when,
max_modification_date);
            System.out.println("Modified_when = " + modified_when.toString());
            System.out.println("Generating order items");
            OrderItem [] items = generateOrderItems(orderId, client,
created_when);
            Error [] errors = generateErrors(client, orderId, created_when);

            return new Order(orderId,
                    customer_id,
                    status,
                    created_when,
                    modified_when,
                    items,
```

```
                    errors
            );
        } catch (SQLException e)
        {
            e.printStackTrace();
            return null;
        }
    }

    static OrderItem [] generateOrderItems(long order_id, DBclient client,
Date order_created_when)
    {
        int itemsSize = Random.genRandomInt(MIN_ORDER_ITEMS, MAX_ORDER_ITEMS);
        OrderItem [] result = new OrderItem [itemsSize];
        for (int i = 0; i < itemsSize; i++)
        {
            result [i] = OrderItem.generateOrderItem(order_id, client,
order_created_when);
        }
        return result;
    }

    static Error [] generateErrors(DBclient client, long order_id, Date
order_created_when)
    {
        int itemsSize = Random.genRandomInt(MIN_ERRORS, MAX_ERRORS);
        Error [] result = new Error [itemsSize];
        for (int i = 0; i < itemsSize; i++)
        {
            result [i] = Error.generateError(client, order_id, i,
order_created_when);
        }
        return result;
    }

    static int randomizeStatus(DBclient client) throws SQLException
    {
        String query = "select MAX(status_id) from order_status;";
        int maxStatusId = client.selectMaxStatusId();  // todo: zero exception
```

```
        return Random.genRandomInt(MIN_STATUS_ID, maxStatusId);
    }

    static Exception generateException()
    {
        try {
            throw new NullPointerException();
        }
        catch (Exception e)

        {
            return e;
        }
    }
}
```

Класс OrderItem:

```
import java.sql.SQLException;
import java.util.Date;

class OrderItem
{
    static final long MIN_ITEM_ID = 1000000001;
    static final int MAX_QUANTITY = 4;
    static final int MIN_QUANTITY = 1;
    static final long INITIAL_ID = 30000000;

    long order_id;
    long item_id;
    int quantity;
    Date created_when;

    private OrderItem(long order_id,
                      long item_id,
                      int quantity,
                        Date created_when
            )
    {
```

```
        this.order_id = order_id;
        this.item_id = item_id;
        this.quantity = quantity;
        this.created_when = created_when;
    }

    static OrderItem generateOrderItem(long order_id, DBclient client, Date
order_created_when)
    {
        System.out.println("Getting a max item id");
        try {
            long maxItemId = client.selectMaxItemId();
            if (maxItemId == 0)
            {
                maxItemId = INITIAL_ID;
            }
            System.out.println("Max item id  = " + maxItemId);
            System.out.println("Randomizing an item id");
            long itemId = Random.genRandomLong(MIN_ITEM_ID, maxItemId);
            int quantity = Random.genRandomInt(MIN_QUANTITY, MAX_QUANTITY);
            return new OrderItem(order_id, itemId, quantity,
order_created_when);
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }

    static Exception generateException()
    {
        try {
            throw new ClassNotFoundException();
        }
        catch (Exception e)
        {
            return e;
        }
    }
}
```

Класс Email:

```
import java.util.Date;

class Email
{
    long customer_id;
    String email;
    Date created_when;

    Email(long customer_id,
          String email,
          Date created_when)
    {
        this.customer_id = customer_id;
        this.email = email;
        this.created_when = created_when;
    }

    static Exception generateException()
    {
        try {
            throw new IndexOutOfBoundsException();
        }
        catch (Exception e)

        {
            return e;
        }
    }
}
```

Класс Address:

```
import java.util.Date;

class Address
{
    long customer_id;
    String street_and_building;
```

```
    String city;
    String country;
    String zip;
    String phone;
    Date created_when;

    Address(long customer_id,
            String street_and_building,
            String city,
            String country,
            String zip,
            String phone,
            Date created_when
    )
    {
        this.customer_id = customer_id;
        this.street_and_building = street_and_building;
        this.city = city;
        this.country = country;
        this.zip = zip;
        this.phone = phone;
        this.created_when = created_when;
    }

    static Exception generateException()
    {
        try {
            throw new RuntimeException();
        }
        catch (Exception e)

        {
            return e;
        }
    }
}
```

Класс Customer:

```
import java.io.*;
import java.sql.SQLException;
import java.util.Date;
import java.util.GregorianCalendar;

class Customer
{
    static final long INITIAL_ID = 2000000001;
    static final long CUSTOMER_ID_MASK = INITIAL_ID;
    static final int MAX_ORDERS = 12;
    static final int MIN_ORDERS = 1;

    long customer_id;
    String name;
    String surname;
    Address address;
    Email email;
    Date created_when;
    Order [] orders;

    private Customer(long customer_id,
              String name,
              String surname,
              Address address,
              Email email,
              Date created_when,
              Order [] orders
    )
    {
        this.customer_id = customer_id;
        this.name = name;
        this.surname = surname;
        this.address = address;
        this.email = email;
        this.created_when = created_when;
        this.orders = orders;
    }
```

```
    static Customer nextSampleCustomer(DBclient client)
    {
        System.out.println("Getting a max customer id");

        long maxCustomerId = 0;
        try {
            maxCustomerId = client.selectMaxCustomerId();
            if (maxCustomerId == 0)
            {
                maxCustomerId = INITIAL_ID;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }

        System.out.println("Max customer id = " + maxCustomerId);

        System.out.println("Opening sample data file");
        File sampleDataFile = new
File("C:/Users/Admvoute/IdeaProjects/NIR_test_data_generator/src/customers_sample_d
ata.csv");
        BufferedReader reader = null;

        int lineToRead = (int)(maxCustomerId - CUSTOMER_ID_MASK + 1);

        try {
            reader = new BufferedReader(new FileReader(sampleDataFile));
            System.out.println("Reading a line number = " + lineToRead);
            String line;

            int n = 0;
            while ((line = reader.readLine()) != null && n < lineToRead) {
                n++;
            }

            System.out.println("Customer is found");
            String [] customerFields = line.split(";");  //todo: null exception
```

```
//                  CSV map:
"first_name","last_name","company_name","address","city","county","state","zip","ph
one1","phone2","email","web"
            System.out.println(" [name: " + customerFields [0] + ", lastname: " +
customerFields [1] + " ... ]");

            long customer_id = CUSTOMER_ID_MASK + lineToRead;
            Date created_when =
Random.genRandomDate(NIR_test_data_generator.OBJECTS_START_DATE,
                    NIR_test_data_generator.current_date);

            System.out.println("Creating  an address");
            Address address = new Address(customer_id,
                    customerFields [3],
                    customerFields [4],
                    customerFields [6],
                    customerFields [7],
                    customerFields [8],
                    created_when
            );
//                System.out.println("Address is " + address.toString());
            System.out.println("Creating a email");
            Email email = new Email(customer_id, customerFields [10],
created_when);
//                  System.out.println("Email is " + email.toString());
            Order [] orders = generateOrders(client, customer_id, created_when);

            return new Customer(customer_id,
                    customerFields [0],
                    customerFields [1],
                    address,
                    email,
                    created_when,
                    orders
            );
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        return null;
    }

    static Order [] generateOrders(DBclient client, long customer_id, Date
customer_created_when)
    {
        int itemsSize = Random.genRandomInt(MIN_ORDERS, MAX_ORDERS);
        Order [] result = new Order [itemsSize];
        for (int i = 0; i < itemsSize; i++)
        {
            result [i] = Order.generateOrder(customer_id, customer_created_when,
client, i);

        }
        return result;
    }

    static Exception generateException()
    {
        try {
            throw new IllegalArgumentException();
        }
        catch (Exception e)

        {
            return e;
        }
    }
}
```

Класс Random:

```
import java.util.Date;

final class Random {
    static long genRandomLong(long min, long max)
    {
        return (long)( (Math.random() * (max - min)) + min);
    }
```

```
    static int genRandomInt(int min, int max)
    {
        return (int)( (Math.random() * (max - min)) + min);
    }
    static Date genRandomDate(Date minDate, Date maxDate)
    {
        long max = maxDate.getTime();
        long min = minDate.getTime();
        return new Date( (long)( (Math.random() * ( max - min )) + min));
    }}
```

ПРИЛОЖЕНИЕ В

Код Javascript утилиты Indices Join Tool

Скрипт es_join_tool.js.

```
var express = require('express');
var body_parser = require('body-parser');
var routes = require('./routes.js');
var app = express();

app.use(body_parser.json({ extended: true }));

routes(app);

var server = app.listen(3560, function() {
    console.log('app running', server.address().port);
});
```

Скрипт routes.js.

```
var es = require('elasticsearch');
var initiateLogger = require('./logger.js');

  const logger = initiateLogger.getLogger('ijs_main');
  logger.debug('Entering cheese testing');

var client = new es.Client(
    {hosts: [
        'http://localhost:9210'
    ]}
);

var indexCommand = {
    index:  { _index: null, _type: null }
}

var appRouter = function (app) {
    app.post('/test', function(req, res) {
        console.log('post test request is recieved');
```

```
            console.log(req.body.field1);
            res.status(200).send('test response');
            logger.debug('Entering cheese testing');
        });

        app.post('/readData', function(req, resp) {
            console.log('Request readData is received.');
            if (req.body) {
                let reqBody = req.body;
                let reqIndex, reqID;
                if (reqBody.index) {
                    reqIndex = reqBody.index;
                }
                if (reqBody.id) {
                    reqID = reqBody.id;
                }
                if (reqIndex && reqID) {
                    console.log('Asking ES with following parameters: index: %s,
id: %s', reqIndex, reqID);
                    client.get({
                        index: reqIndex,
                        type: reqIndex,
                        id: reqID
                    }, function(err, esResp, status) {
                        if (err) {
                            console.log(err);
                        } else {
                            console.log(esResp);
                            resp.status(200).send(esResp);
                        }
                    });
                    // console.log(esResponse);
                }
            }


        });

        app.post('/join_to_new', function(commandReq, commandResp) {
```

```
            logger.trace('join_to_new    is    requested,    request    body:%s',
JSON.stringify(commandReq.body));

            let userResponse = {
                result: null,
                source1_count: 0,
                source2_count: 0
            }

            let reqBody = commandReq.body;
            if (reqBody.target.name) {
                indexCommand.index._index = reqBody.target.name;
            } else {
                userResponse.result = 'target.name is not specified';
                commandResp.status(400).send(userResponse);
                return;
            }

            let source1 = {
                index: reqBody.source[0].index,
                query: { query: reqBody.source[0].query},
                join_field: reqBody.source[0].join_field,
                join_field_unique: reqBody.source[0].join_field_unique,
                fields: reqBody.source[0].fields,
                count: 0
            }
            source1.fields.push(source1.join_field);

            let source2 = {
                index: reqBody.source[1].index,
                query: { query: reqBody.source[1].query},
                join_field: reqBody.source[1].join_field,
                join_field_unique: reqBody.source[1].join_field_unique,
                fields: reqBody.source[1].fields,
                count: 0
            }
            source2.fields.push(source2.join_field);
            let curDate = new Date();
            indexCommand.index._type   =   'join_'   +   source1.index   +   '_'   +
source2.index +
```

```
            curDate.getFullYear() +
            curDate.getMonth().toLocaleString() +
            curDate.getDay().toLocaleString();

            if (reqBody.source[0].join_field_unique) {
                sourceUniqueKeysConfig = source1;
                sourceTargetConfig = source2;
            } else if (reqBody.source[1].join_field_unique) {
                sourceUniqueKeysConfig = source2;
                sourceTargetConfig = source1;
            } else {
                userResponse.result = 'There is no source with join_field_unique
specified';
                commandResp.status(400).send(userResponse);
                logger.error(userResponse.result);
                return;
            }

            logger.trace('Checking source1 hits');
            client.count(
                {
                    index: source1.index,
                    body: source1.query
                }
            ).then(function(countResponse1) {
                if (countResponse1.error) {
                    userResponse.result = 'Error in getting count for source1';
                    userResponse.error = countResponse1;
                    commandResp.status(400).send(userResponse);
                    logger.error(countResponse1);
                } else {
                    logger.trace('Source1 hits: %s', countResponse1.count);
                    userResponse.source1_count = countResponse1.count;

                    logger.trace('Checking source2 hits');
                    client.count(
                        {
                            index: source2.index,
                            body: source2.query
                        }
```

```
                    ).then(function(countResponse2) {
                        if (countResponse2.error) {
                            userResponse.result = 'Error in getting count for
source2';
                            userResponse.error = countResponse2;
                            commandResp.status(400).send(userResponse);
                            logger.error(countResponse2);
                        } else {
                            logger.trace('Source2 hits: %s',
countResponse2.count);
                            userResponse.source2_count = countResponse2.count;

                            if (countResponse1.count > 0 && countResponse1.count >
0) {
                                userResponse.result = 'Join is started';
                                commandResp.status(200).send(userResponse);
                                source1.count = countResponse1.count;
                                source2.count = countResponse2.count;
                                logger.trace(userResponse.result);
                                createJoinedIndex();
                            } else {
                                userResponse.result = 'One source is empty,
nothing to join';
                                commandResp.status(200).send(userResponse);
                                logger.trace(userResponse.result);
                            }
                        }
                    });
                }
            });

        });

    }

    var sourceUniqueKeysConfig;
    var sourceTargetConfig;

    var createJoinedIndex = function(){
```

```
        logger.trace('Joining the indices');

        logger.trace('Starting scrolling');
        client.search({
            index: sourceUniqueKeysConfig.index,
            scroll: '20s', // keep the search results "scrollable" for 30 seconds
            _sourceInclude: sourceUniqueKeysConfig.fields, // filter the source to
only include the title field
            body: sourceUniqueKeysConfig.query,
            size:500
        }, proceedScrollData);

    }

    var sourceUniqueKeysDocs = new Map();
    var joinedDocsBulkInserts = [];
    var totalDocsScrolled = 0;
    var scroll_id;

    var proceedScrollData = function (error, response) {
        if (error) {
            logger.error(error);
        } else {
            logger.trace('Documents scrolled: %s', response.hits.hits.length);
            totalDocsScrolled += response.hits.hits.length;
            logger.trace('Total docs scrolled: % s', totalDocsScrolled);
            // if (target) {
            response.hits.hits.forEach(function (hit) {

sourceUniqueKeysDocs.set(hit._source[sourceUniqueKeysConfig.join_field_unique],
hit._source);
            });
            logger.trace('Source docs are mapped with join fields as keys');
            let targetTerms = Array.from(sourceUniqueKeysDocs.keys());
            // let targetHits =
            scroll_id = response._scroll_id;

            response.hits.total;
            searchForTerms(sourceTargetConfig, targetTerms);
        }
```

```
    }

    var proceedScrolling = function() {

        logger.trace('Proceeeding scrolling');
        if (totalDocsScrolled < sourceUniqueKeysConfig.count) {
            // now we can call scroll over and over
            client.scroll({
              scrollId: scroll_id,
              scroll: '20s'
            }, proceedScrollData);
        } else {
            logger.trace('all done: required records %s, hits processed %s',
sourceUniqueKeysConfig.count, totalDocsScrolled);
        }
    }

    var joinResults = function(targetHits) {
        logger.trace('Joining the surce docs with target docs');
        targetHits.forEach(function (targetHit) {
            let                              sourceHit                              =
sourceUniqueKeysDocs.get(targetHit._source[sourceTargetConfig.join_field]);
            Object.keys(sourceHit).forEach(key => {
                let fieldName = key;
                if (targetHit._source[key]) {
                    fieldName = key + '_joined';
                    let n = 0;
                    while (targetHit._source[fieldName]) {
                        n++;
                        fieldName = key + '_joined' + n;
                    }
                }
                targetHit._source[fieldName] = sourceHit[key];
            });
            // logger.trace('index command is' + JSON.stringify(indexCommand));
            joinedDocsBulkInserts.push(indexCommand);
            joinedDocsBulkInserts.push(targetHit._source);
        });
        insertJoinedData(sourceUniqueKeysDocs);
    }
```

```
var insertJoinedData = function(joinedData) {

    logger.trace('Joined docs to be inserted: %s', joinedData.size);
    // logger.trace(JSON.stringify(joinedDocsBulkInserts));
    client.bulk({
        body: joinedDocsBulkInserts
    }, function(err, resp) {
        if (err) {
            logger.error(err);
        } else {
            resp.items = null;
            logger.trace('Docs are indexed, response: %s', resp);
            sourceUniqueKeysDocs.clear();
            proceedScrolling();
        }
    })

}

var searchForTerms = function(target, terms) {
    logger.trace('Searching for terms size: %s', terms.length);

    let query = {
        query: {
            bool: {
                must: [
                    target.query.query,
                    {
                        terms:
                        {
                            [target.join_field]: terms
                        }
                    }
                ]
            }
        }
    }

    client.search({
```

```
            index: target.index,
            _sourceInclude: target.fields, // filter the source to only include
the title field
            body: query,
            size: terms.length
        }, function (error, response) {
            if (error) {
                logger.error(error);
            } else {
                joinResults(response.hits.hits);
            }
        }
        )

    }

    module.exports = appRouter;
```

Скрипт logger.js.

```
var log4js = require('log4js');
var fs = require('fs');

    log4js.configure({
        appenders: { ijs_main: {
            type: 'file',
            filename: './logs/ijs.log',
            maxLogSize: 5242880,    //5mb
            backups: 5
        } },
        categories: { default: { appenders: ['ijs_main'], level: 'trace' } }
    });


module.exports = log4js;
```

ПРИЛОЖЕНИЕ С

Код Javascript утилиты Report tool, предназначенной для сборки отчетов с выходом в виде файла CSV либо email

Скрипт send_email.js:

```
var nodemailer = require('nodemailer');
var es = require('elasticsearch');
var report1 = require('./report1.js');
var report2 = require('./report2.js');
var ps = require('./ps.js');
var fs = require('fs');
var reportParams = {};
var initiateLogger = require('./logger.js');
var logger = initiateLogger.getLogger('report1');

var es_client = new es.Client(
  {hosts: [
      'http://localhost:9210'
  ]}
);

var transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
      user: 'leopardo.jakomo@gmail.com',
      pass: ps
    }
});

var reportAction = function(report) {
  if (reportParams.email === "false") {

    const fs = require('fs');
    fs.writeFile("./" + reportParams.reportName + ".csv", report,
function(err) {
        if(err) {
            return logger.log(err);
```

```
        }

        console.log("The report1 is saved!");
    });

  } else if (reportParams.email === "true") {
    var mailOptions = {
      from: 'leopardo.jakomo@gmail.com',
      to: 'leopardo.jakomo@gmail.com',
      subject: 'Отчет',
      html: report
    };

    transporter.sendMail(mailOptions, function(error, info){
      if (error) {
        logger.log(error);
      } else {
        logger.log('Email sent: ' + info.response);
      }
    });
  }
}

var generateReport = function(client) {

  reportParams.reportName = process.argv [2];
  reportParams.email = process.argv [3];
  reportParams.startDate = process.argv [4];
  reportParams.endDate = process.argv [5];

  for (let j = 0; j < process.argv.length; j++) {
    logger.log(j + ' -> ' + (process.argv [j]));
  }

  if (reportParams.reportName === 'report1') {
     return report1(client, reportAction, reportParams);
  } else if (reportParams.reportName === 'report2') {
    return report2(client, reportAction, reportParams);
  }
```

```
}

generateReport(es_client);
```

Скрипт report1.js:

```
var initiateLogger = require('./logger.js');
var logger = initiateLogger.getLogger('report1');
var Mustache = require('Mustache');
var table_lib = require('./html_table_lib.js');
var currentDate = new Date();
var weekAgoDate = new Date();
weekAgoDate.setDate(currentDate.getDate() - 7);
var endDate = currentDate.getFullYear() + '-' + currentDate.getMonth() + '-' +
currentDate.getDay();
var startDate = weekAgoDate.getFullYear() + '-' + weekAgoDate.getMonth() + '-'
+ weekAgoDate.getDay();

var reportQuery = {
    index: 'order,error',
    queryBody: {
        "size": 0,
        "query": {
          "range": {
            "created_when": {
              "gte": startDate,
              "lte": endDate
            }
          }
        },
        "aggs": {
          "stats": {
            "filters": {
              "filters": {
                "orders_created": {
                  "bool": {
                    "must" : [
                        {
                          "term": {
```

```
                "_type": "order"
              }
            }
          ]
        }
      },
      "orders_completed": {
        "bool": {
          "must": [
            {
              "term": {
                 "_type": "order"
              }
            },
            {
              "term": {
                "status_name.keyword": "Delivered"
              }
            }
          ]
        }
      },
      "err_created": {
        "bool": {
          "must": [
            {
              "term": {
                 "_type": "error"
              }
            }
          ]
        }
      },
      "orders_completed_witihin_5_days": {
        "bool": {
          "must": [
            {
              "term": {
                 "_type": "order"
```

```
                        }
                      },
                      {
                        "term": {
                          "status_name.keyword": "Delivered"
                        }
                      },
                      {
                        "script": {
                            "script":{
                              "inline":"((doc ['completed_when'].value - doc
['created_when'].value)/1000/60/60/24) <= 5",
                              "lang":"painless"
                          }
                        }
                      }
                    ]
                  }
                }
              }
            },
            "aggs": {
              "unique_orders": {
                "cardinality": {
                  "field": "order_id"
                }
              }
            }
          },
          "avg_orders_created_per_day": {
            "avg_bucket": {
              "buckets_path": "orders_created_per_day>orders_created"
            }
          },
          "orders_created_per_day": {
            "date_histogram": {
              "field": "created_when",
              "interval": "1d",
              "min_doc_count": 1
```

```
            },
            "aggs": {
              "orders_created": {
                "cardinality": {
                  "field": "order_id"
                }
              }
            }
          }
        }
      }
}

var mustacheEmail =
//'Наименование показателя,,Цель,,Значение,,Отклонение от цели' +
'{{#aggregations}}' +
    '{{#stats.buckets}}' +
        '1. Количество созданных заказов,,,,' +
'{{orders_created.doc_count}},,;;' +
        '2. Процент выполненных заказов,,' +
            '{{aim2}},,' +
            '{{#calcRatio}}' +
                '{{orders_completed.doc_count}},{{orders_created.doc_count}}'
+
            '{{/calcRatio}},,' +
            '{{#stats.buckets.calcDiff}}' +
                '{{aim2}},' +
                '{{#calcRatio}}' +

'{{orders_completed.doc_count}},{{orders_created.doc_count}}' +
                '{{/calcRatio}},,' +
            '{{/stats.buckets.calcDiff}};;' +
        '3. Процент выполненных заказов в течение 5 дней,,' +
            '{{aim3}},,' +
            '{{#calcRatio}}' +

'{{orders_completed_witihin_5_days.doc_count}},{{orders_completed.doc_count}}'
+
            '{{/calcRatio}},,' +
```

```
            '{{#stats.buckets.calcDiff}}' +
                '{{aim3}},' +
                '{{#calcRatio}}' +

'{{orders_completed_witihin_5_days.doc_count}},{{orders_completed.doc_count}}'
+
                '{{/calcRatio}},,' +
            '{{/stats.buckets.calcDiff}};;' +
        '4. Количество проблемных заказов,,,,' +
            '{{err_created.unique_orders.value}},,;;' +
        '5. Процент проблемных заказов,,'+
            '{{aim5}},,' +
            '{{#calcRatio}}' +

'{{err_created.unique_orders.value}},{{orders_created.doc_count}}' +
            '{{/calcRatio}},,' +
            '{{#stats.buckets.calcDiff}}' +
                '{{aim5}},' +
                '{{#calcRatio}}' +

'{{err_created.unique_orders.value}},{{orders_created.doc_count}}' +
                '{{/calcRatio}},,' +
            '{{/stats.buckets.calcDiff}};;' +
    '{{/stats.buckets}}' +

    '6. Среднее количество заказов создаваемых ежедневно,,'+
        '{{stats.buckets.aim6}},,' +
        '{{#stats.buckets.round}}' +
            '{{avg_orders_created_per_day.value}}' +
        '{{/stats.buckets.round}},,' +
        '{{#stats.buckets.calcDiff}}' +
            '{{stats.buckets.aim6}},{{avg_orders_created_per_day.value}}' +
        '{{/stats.buckets.calcDiff}};;' +

'{{/aggregations}}'
;

var mustacheFile =
'Наименование показателя,Цель,Значение,Отклонение от цели\n' +
```

```
'{{#aggregations}}' +
    '{{#stats.buckets}}' +
        '1. Количество созданных заказов,,' +
'{{orders_created.doc_count}},\n' +
        '2. Процент выполненных заказов,' +
            '{{aim2}},' +
            '{{#calcRatio}}' +
                '{{orders_completed.doc_count}},{{orders_created.doc_count}}'
+
            '{{/calcRatio}},' +
            '{{#stats.buckets.calcDiff}}' +
                '{{aim2}},' +
                '{{#calcRatio}}' +

'{{orders_completed.doc_count}},{{orders_created.doc_count}}' +
                '{{/calcRatio}},' +
            '{{/stats.buckets.calcDiff}}\n' +
        '3. Процент выполненных заказов в течение 5 дней,' +
            '{{aim3}},' +
            '{{#calcRatio}}' +

'{{orders_completed_witihin_5_days.doc_count}},{{orders_completed.doc_count}}'
+
            '{{/calcRatio}},' +
            '{{#stats.buckets.calcDiff}}' +
                '{{aim3}},' +
                '{{#calcRatio}}' +

'{{orders_completed_witihin_5_days.doc_count}},{{orders_completed.doc_count}}'
+
                '{{/calcRatio}},' +
            '{{/stats.buckets.calcDiff}}\n' +
        '4. Количество проблемных заказов,,' +
            '{{err_created.unique_orders.value}},\n' +
        '5. Процент проблемных заказов,'+
            '{{aim5}},' +
            '{{#calcRatio}}' +

'{{err_created.unique_orders.value}},{{orders_created.doc_count}}' +
```

```
            '{{/calcRatio}},' +
            '{{#stats.buckets.calcDiff}}' +
                '{{aim5}},' +
                '{{#calcRatio}}' +

'{{err_created.unique_orders.value}},{{orders_created.doc_count}}' +
                '{{/calcRatio}},' +
            '{{/stats.buckets.calcDiff}}\n' +
    '{{/stats.buckets}}' +

    '6. Среднее количество заказов, создаваемых ежедневно,'+
        '{{stats.buckets.aim6}},' +
        '{{#stats.buckets.round}}' +
            '{{avg_orders_created_per_day.value}}' +
        '{{/stats.buckets.round}},' +
        '{{#stats.buckets.calcDiff}}' +
            '{{stats.buckets.aim6}},{{avg_orders_created_per_day.value}}' +
        '{{/stats.buckets.calcDiff}}\n' +

'{{/aggregations}}'
;

var getReport = function(client, action, params) {

    if (params.startDate !== null && params.endDate !== null) {
        startDate = params.startDate;
        endDate = params.endDate;
        reportQuery.queryBody.query.range.created_when.gte = params.startDate;
        reportQuery.queryBody.query.range.created_when.lte = params.endDate;
    }

    logger.debug(params.startDate);
    logger.debug(params.endDate);

    client.search({
        index: reportQuery.index,
        body: reportQuery.queryBody,
    }, function (error, response) {
        if (error) {
```

```
            logger.error(error);
        } else {
            response.aggregations.stats.buckets.aim2 = 0.90;
            response.aggregations.stats.buckets.aim3 = 0.95;
            response.aggregations.stats.buckets.aim5 = 0.05;
            response.aggregations.stats.buckets.aim6 = 10;

            response.aggregations.stats.buckets.calcRatio = function () {
                return function (text, render) {
                    let params = render(text).split(',');
                    return (params [0] / params [1]).toFixed(2);
                }
            }
            response.aggregations.stats.buckets.calcDiff = function () {
                return function (text, render) {
                    let params = render(text).split(',');
                    return (Math.abs(params [0] - params [1])).toFixed(2);
                }
            }
            response.aggregations.stats.buckets.round = function () {
                return function (text, render) {
                    let data = render(text);
                    return Number.parseFloat(data).toFixed(2);
                }
            }
            logger.debug(JSON.stringify(response));
            var rendered;

            logger.debug(params.email);

            if (params.email === "false") {
                rendered = Mustache.render(mustacheFile, response);
                action(rendered);
            } else if (params.email === "true") {
                rendered = Mustache.render(mustacheEmail, response);
                action(generateEmail(rendered));
            }

        }
```

```
    }
    )

}

var headingValues = [
    "Наименование показателя",
    "Цель",
    "Значение",
    "Отклонение от цели"
  ];

  var columnsTDWidths = [
    '300',
    '90',
    '90',
    '90'
  ];

  function buildParams()
  {
    let tableParams = table_lib.getDefaultParamsMap();
    let pFontStyle =
tableParams.get(table_lib.OBJECT_P).get(table_lib.PARAM_STYLE);
    pFontStyle.set("font-family", "Times New Roman");

    let trHeadingFontStyle =
tableParams.get(table_lib.OBJECT_TR_HEADING).get(table_lib.PARAM_STYLE);
    trHeadingFontStyle.set(table_lib.STYLE_WEIGHT, "bold");
    let tdHeadingFontStyle =
tableParams.get(table_lib.OBJECT_TD_HEADING).get(table_lib.PARAM_STYLE);
    tdHeadingFontStyle.delete("background");
    let pHeadingFontStyle =
tableParams.get(table_lib.OBJECT_P_HEADING).get(table_lib.PARAM_STYLE);
    pHeadingFontStyle.set("font-family", "Times New Roman");

    let tdParamSets = [];

    for (let i = 0; i < columnsTDWidths.length; i++)
```

```
    {
      let tdParams = table_lib.getDefaultParamsMap().get(table_lib.OBJECT_TD);
      let tdStyle = tdParams.get(table_lib.PARAM_STYLE);
      tdStyle.set("width", columnsTDWidths [i]);
      tdStyle.set("valign", "top");
      tdParamSets [i] = tdParams;
    }

    let conditions = tableParams.get(table_lib.OBJECT_CONDITIONS);

    conditions.set(table_lib.OBJECT_TD, function(data, index)
    {
      if (index < tdParamSets.length) return tdParamSets [index];
      return null;
    });

    return tableParams;
  }

var generateEmail = function(data) {
    let introduction = "Добрый день," +
    "<p>"+
    "отчет показателей по заказам за период: " +
    startDate + " - " + endDate +
    "<p>"
    ;
    let tableParams = buildParams();
    let tableData = table_lib.buildMap(data, ";;", ",,");
    return introduction + table_lib.buildTable(headingValues, tableData,
tableParams);
}

module.exports = getReport;
```

Скрипт report2.js:

```
var initiateLogger = require('./logger.js');
var logger = initiateLogger.getLogger('report2');
var Mustache = require('Mustache');
```

```
var table_lib = require('./html_table_lib.js');
var currentDate = new Date();
var weekAgoDate = new Date();
weekAgoDate.setDate(currentDate.getDate() - 7);
var endDate = currentDate.getFullYear() + '-' + currentDate.getMonth() + '-' +
currentDate.getDay();
var startDate = weekAgoDate.getFullYear() + '-' + weekAgoDate.getMonth() + '-'
+ weekAgoDate.getDay();

var reportQuery = {
    index: 'order,error',
    queryBody: {
        "size": 0,
        "query": {
          "bool": {
            "must": [
              {
                "range": {
                  "created_when": {
                     "gte": startDate,
                     "lte": endDate
                  }
                }
              }
            ]
          }
        },
        "aggs": {
          "errors": {
            "terms": {
              "field": "class.keyword",
              "size": 10
            },
            "aggs": {
              "orders_count": {
                "cardinality": {
                  "field": "order_id"
                }
              }
```

```
            }
          }
        }
      }
}

var mustacheEmail =
//'Класс ошибки,Количество ошибок,Количество проблемных заказов\n' +
'{{#aggregations}}' +
    '{{#errors.buckets}}' +
        '{{key}},,' +
        '{{doc_count}},,' +
        '{{orders_count.value}};;' +
    '{{/errors.buckets}}' +
'{{/aggregations}}'
;

var mustacheFile =
'Класс ошибки,Количество ошибок,Количество проблемных заказов\n' +
'{{#aggregations}}' +
    '{{#errors.buckets}}' +
        '{{key}},' +
        '{{doc_count}},' +
        '{{orders_count.value}}\n' +
    '{{/errors.buckets}}' +
'{{/aggregations}}'
;

var getReport = function(client, action, params) {

    if (params.startDate !== null && params.endDate !== null) {
        startDate = params.startDate;
        endDate = params.endDate;
        reportQuery.queryBody.query.bool.must [0].range.created_when.gte =
params.startDate;
        reportQuery.queryBody.query.bool.must [0].range.created_when.lte =
params.endDate;
    }
```

```
    logger.debug(params.startDate);
    logger.debug(params.endDate);

    client.search({
        index: reportQuery.index,
        body: reportQuery.queryBody,
    }, function (error, response) {
        if (error) {
            logger.error(error);
        } else {
            logger.debug(JSON.stringify(response));
            var rendered;

            logger.debug(params.email);

            if (params.email === "false") {
                rendered = Mustache.render(mustacheFile, response);
                action(rendered);
            } else if (params.email === "true") {
                rendered = Mustache.render(mustacheEmail, response);
                action(generateEmail(rendered));
            }

        }
    }
    )

}

var headingValues = [
    "Класс ошибки",
    "Количество ошибок",
    "Количество проблемных заказов"
  ];

  var columnsTDWidths = [
    '90',
    '90',
    '90'
```

```
  ];

  function buildParams()
  {
    let tableParams = table_lib.getDefaultParamsMap();
    let pFontStyle =
tableParams.get(table_lib.OBJECT_P).get(table_lib.PARAM_STYLE);
    pFontStyle.set("font-family", "Times New Roman");

    let trHeadingFontStyle =
tableParams.get(table_lib.OBJECT_TR_HEADING).get(table_lib.PARAM_STYLE);
    trHeadingFontStyle.set(table_lib.STYLE_WEIGHT, "bold");
    let tdHeadingFontStyle =
tableParams.get(table_lib.OBJECT_TD_HEADING).get(table_lib.PARAM_STYLE);
    tdHeadingFontStyle.delete("background");
    let pHeadingFontStyle =
tableParams.get(table_lib.OBJECT_P_HEADING).get(table_lib.PARAM_STYLE);
    pHeadingFontStyle.set("font-family", "Times New Roman");

    let tdParamSets = [];

    for (let i = 0; i < columnsTDWidths.length; i++)
    {
      let tdParams = table_lib.getDefaultParamsMap().get(table_lib.OBJECT_TD);
      let tdStyle = tdParams.get(table_lib.PARAM_STYLE);
      tdStyle.set("width", columnsTDWidths [i]);
      tdStyle.set("valign", "top");
      tdParamSets [i] = tdParams;
    }

    let conditions = tableParams.get(table_lib.OBJECT_CONDITIONS);

    conditions.set(table_lib.OBJECT_TD, function(data, index)
    {
      if (index < tdParamSets.length) return tdParamSets [index];
      return null;
    });

    return tableParams;
```

```
  }

var generateEmail = function(data) {
    let introduction = "Добрый день," +
    "<p>"+
    "список ошибок выполнения заказов за период: " +
    startDate + " - " + endDate +
    "<p>"
    ;
    let tableParams = buildParams();
    let tableData = table_lib.buildMap(data, ";;", ",,");
    return introduction + table_lib.buildTable(headingValues, tableData,
tableParams);
}

module.exports = getReport;
```

Скрипт logger.js:

```
var log4js = require('log4js');
var fs = require('fs');

    log4js.configure({
        appenders: { report1: {
            type: 'file',
            filename: './logs/report1.log',
            maxLogSize: 5242880,    //5mb
            backups: 5
        },
            report2: {
                  type: 'file',
                  filename: './logs/report2.log',
                  maxLogSize: 5242880,    //5mb
                  backups: 5
            }           },
        categories: { default: { appenders: ['report1'], level: 'trace' } }
    });

module.exports = log4js;
```

Скрипт html_table_lib.js:

```
const OBJECT_TABLE = "table";
const OBJECT_TR_HEADING = "tr_heading";
const OBJECT_TD_HEADING = "td_heading";
const OBJECT_P_HEADING = "p_heading";
const OBJECT_TR = "tr";
const OBJECT_TD = "td";
const OBJECT_P = "p";
const OBJECT_CONDITIONS = "formattingConditions";
const NO_HEADING = 'no_heading';
const NO_HEADING_VALUE = false;
const PARAM_STYLE = "style";
const STYLE_COLOR = "color";
const STYLE_WEIGHT = "font-weight"; // normal or bold

/* created by egto1016 */

function getDefaultParamsMap()
{
  let tableDefaultStyleParamsMap = new Map( [
    ["border-collapse","collapse"],
    ["border","none"],
    ["mso-border-alt","solid windowtext .5pt"],
    ["mso-yfti-tbllook","1184"],
    ["mso-padding-alt","0cm 5.4pt 0cm 5.4pt"]
  ]);
  let tableDefaultParamsMap = new Map( [
    ["class", "MsoTableGrid"],
    ["cellspacing", "0"],
    ["cellpadding", "0"],
    [PARAM_STYLE, tableDefaultStyleParamsMap]
  ]);

  // HEADING
  // TR HEADING
  let trHeadingDefaultStyleParamsMap = new Map( [
    ["mso-yfti-irow","0"],
    ["mso-yfti-firstrow","yes"],
```

```
    ["height","15.0pt"]
  ]);
  let trHeadingDefaultParamsMap = new Map( [
    [PARAM_STYLE, trHeadingDefaultStyleParamsMap]
  ]);

  // TD HEADING
  let tdHeadingDefaultStyleParamsMap = new Map( [
      ["border","solid windowtext 1.0pt"],
      ["mso-border-alt","solid windowtext .5pt"],
      ["background","#C6D9F1"],
      ["mso-background-themecolor","text2"],
      ["mso-background-themetint","51"],
      ["padding","0cm 5.4pt 0cm 5.4pt"],
      ["height","15.0pt"]
  ]);
  let tdHeadingDefaultParamsMap = new Map( [
    ["valign","center"],
    [PARAM_STYLE,tdHeadingDefaultStyleParamsMap]
  ]);

  // P HEADING
  let pHeadingDefaultStyleParamsMap = new Map( [

  ]);
  let pHeadingDefaultParamsMap = new Map( [
    ["class","MsoNormal"],
    [PARAM_STYLE,pHeadingDefaultStyleParamsMap]
  ]);

  // HEADING END

  let trDefaultStyleParamsMap = new Map( [
    ["mso-yfti-irow","1"],
    ["height","15.0pt"]
  ]);
  let trDefaultParamsMap = new Map( [
    [PARAM_STYLE, trDefaultStyleParamsMap]
  ]);
```

```
  let tdDefaultStyleParamsMap = new Map( [
      ["border","solid windowtext 1.0pt"],
      ["border-top","none"],
      ["mso-border-alt","solid windowtext .5pt"],
      ["mso-border-top-alt","solid windowtext .5pt"],
      ["padding","0cm 5.4pt 0cm 5.4pt"],
      ["height","15.0pt"]
  ]);
  let tdDefaultParamsMap = new Map( [
    ["valign","top"],
    [PARAM_STYLE,tdDefaultStyleParamsMap]
  ]);

  let pDefaultStyleParamsMap = new Map( [

  ]);

  let pDefaultParamsMap = new Map( [
    ["class","MsoNormal"],
    [PARAM_STYLE, pDefaultStyleParamsMap]
  ]);

  let emptyFormattingConditions = new Map( [
    ["tr", null],    // where null will be a function(data, defaultParams)
that returns a paramsMap for object where data is a single value or 2-
dimension array(depends from object) and defaultParams = is map with params,
that should be returned by default
    ["td", null],
    ["p", null],
    [OBJECT_TR_HEADING, null],    // where null will be a function(data,
defaultParams) that returns a paramsMap for object where data is a single
value or 2-dimension array(depends from object) and defaultParams = is map
with params, that should be returned by default
    [OBJECT_TD_HEADING, null],
    [OBJECT_P_HEADING, null]
  ]);

  let defaultParamsMap = new Map( [
```

```
    [OBJECT_TABLE, tableDefaultParamsMap],
    [OBJECT_TR_HEADING, trHeadingDefaultParamsMap],
    [OBJECT_TD_HEADING,tdHeadingDefaultParamsMap],
    [OBJECT_P_HEADING, pHeadingDefaultParamsMap],
    [OBJECT_TR, trDefaultParamsMap],
    [OBJECT_TD, tdDefaultParamsMap],
    [OBJECT_P, pDefaultParamsMap],
    [OBJECT_CONDITIONS, emptyFormattingConditions],
    [NO_HEADING, NO_HEADING_VALUE]
  ]);

  return defaultParamsMap;
}

function buildMap(data, lineSeparator, fieldSeparator)
{
  let rows = data.split(lineSeparator);
  for (let i=0; i < rows.length; i++) {
    rows [i] = rows [i].split(fieldSeparator);
  }
  if (rows [rows.length-1] == "") rows.splice(rows.length-1, 1);
  return rows;
}

function buildTable(headingValues, values, tableParamsMap)
{
//  console.log('Let\'s build a beautiful HTML report table');
//  console.log('table_params_map= [%s]', tableParamsMap.toString());
  if (tableParamsMap == null)
  {
    tableParamsMap = getDefaultParamsMap();
  //  console.log('There are no tableParams, so default params will be
used:');
  }

  let start = "<table ";
  let params = buildObjectParams(tableParamsMap.get(OBJECT_TABLE));
  let _start = ">";
  let heading = '';
```

```
  if (tableParamsMap.get(NO_HEADING) == false) {
    heading = buildHeading(headingValues, tableParamsMap);
  }
  let rows = "";
  for (let i = 0; i < values.length; i++) {
    let rowValues = values [i];
    rows += buildTR(rowValues, tableParamsMap, i);
  }
  let end = "</table>";

  let returnString = [start,
    params,
    _start,
    heading,
    rows,
    end
    ].join('');

//  console.log('table_params= [%s]', params);
//  console.log('The HTML table is ready to be used');
  return returnString;
}

function buildHeading(values, tableParams)
{
//  console.log('Building a heading');
  let rowParams = tableParams.get(OBJECT_TR_HEADING);
  let cellParams = tableParams.get(OBJECT_TD_HEADING);
  let pParams = tableParams.get(OBJECT_P_HEADING);
  let conditions = tableParams.get(OBJECT_CONDITIONS);

  let headingParams = getDefaultParamsMap();
  headingParams.set(OBJECT_TR, rowParams);
  headingParams.set(OBJECT_TD, cellParams);
  headingParams.set(OBJECT_P, pParams);
  headingParams.get(OBJECT_CONDITIONS).set(OBJECT_P,
conditions.get(OBJECT_P_HEADING));
  headingParams.get(OBJECT_CONDITIONS).set(OBJECT_TD,
conditions.get(OBJECT_TD_HEADING));
```

```
  headingParams.get(OBJECT_CONDITIONS).set(OBJECT_TR,
conditions.get(OBJECT_TR_HEADING));
  let heading = buildTR(values, headingParams);

  return heading;
}

function buildTR(values, tableParams, index)
{
  //  console.log('Building a TR');
    if (values.length == 0) return "";
    let rowParams = resolveObjectFormatting(tableParams, values, OBJECT_TR,
index);

    let start = '<tr ';
    let params = buildObjectParams(rowParams);
    let _start = '>';
    let TDs = "";

    for (let i = 0; i < values.length; i++) {
      TDs += buildTD(values [i], tableParams, i);
    }

    let end = '</tr>';
    let returnString = [start,
      params,
      _start,
      TDs,
      end
      ].join('');

  //  console.log('tr_params= [%s], data= [%s]', params, values);
    return returnString;
}

function buildTD(text, tableParams, index) {
  //  console.log("Building a TD");
    let cellParams = resolveObjectFormatting(tableParams, text, OBJECT_TD,
index);
```

```
    let start = '<td ';
    let params = buildObjectParams(cellParams);
    let _start = '>';
    let pText = buildP(text, tableParams, index);
    let end = '</td>';
    let returnString = [start,
      params,
      _start,
      pText,
      end
      ].join('');

  //  console.log('td_params= [%s], data= [%s]', params, text);
    return returnString;
}

function buildP(text, tableParams, index) {
  //  console.log("Building a P");
    let pParams = resolveObjectFormatting(tableParams, text, OBJECT_P, index);

    let start = '<p ';
    let params = buildObjectParams(pParams);
    let _start = '>';
    let end = '</p>';
    let returnString = [start,
      params,
      _start,
      text,
      end
      ].join('');

  //  console.log('p_params= [%s], data= [%s]', params, text);
    return returnString;
}

function resolveObjectFormatting(tableParams, data, objectType, index)
{
  let defaultParams = tableParams.get(objectType);
```

```
  let conditionFunction = tableParams.get(OBJECT_CONDITIONS).get(objectType);
  if (conditionFunction == null)
  {
    return defaultParams;
  } else {
    let formatting = conditionFunction(data, index);
    if (formatting == null) return defaultParams;
    else return formatting;
  }
}

function buildObjectParams(objectParamsMap)
{
  let result = "";

  for (let [objectParam, paramValue] of objectParamsMap)
  {
    if (objectParam.localeCompare(PARAM_STYLE) == 0)
    {
      let styleParams = buildStyleParams(paramValue);
      paramValue = styleParams;
    }
    result += objectParam + "='" + paramValue + "' ";
  }

  return result;
}

function buildStyleParams(styleParamsMap)
{
  let result = "";

  for (let [styleParam, paramValue] of styleParamsMap)
  {
    result += styleParam + ':' + paramValue + ';';
  }
  return result;
}
```

```
exports.buildTable = buildTable;
exports.getDefaultParamsMap = getDefaultParamsMap;
exports.OBJECT_TABLE = OBJECT_TABLE;
exports.OBJECT_TR_HEADING = OBJECT_TR_HEADING;
exports.OBJECT_TD_HEADING = OBJECT_TD_HEADING;
exports.OBJECT_P_HEADING = OBJECT_P_HEADING;
exports.OBJECT_TR = OBJECT_TR;
exports.OBJECT_TD = OBJECT_TD;
exports.OBJECT_P = OBJECT_P;
exports.OBJECT_CONDITIONS = OBJECT_CONDITIONS;
exports.PARAM_STYLE = PARAM_STYLE;
exports.STYLE_COLOR = STYLE_COLOR;
exports.STYLE_WEIGHT = STYLE_WEIGHT;
exports.NO_HEADING = NO_HEADING;
exports.buildMap = buildMap;
```

Скрипт smtp.js (заимствованная библиотека):

```
/* SmtpJS.com - v3.0.0 */
var Email = { send: function (a) { return new Promise(function (n, e) {
a.nocache = Math.floor(1e6 * Math.random() + 1), a.Action = "Send"; var t =
JSON.stringify(a); Email.ajaxPost("https://smtpjs.com/v3/smtpjs.aspx?", t,
function (e) { n(e) }) }) }, ajaxPost: function (e, n, t) { var a =
Email.createCORSRequest("POST", e); a.setRequestHeader("Content-type",
"application/x-www-form-urlencoded"), a.onload = function () { var e =
a.responseText; null != t && t(e) }, a.send(n) }, ajax: function (e, n) { var
t = Email.createCORSRequest("GET", e); t.onload = function () { var e =
t.responseText; null != n && n(e) }, t.send() }, createCORSRequest: function
(e, n) { var t = new XMLHttpRequest; return "withCredentials" in t ? t.open(e,
n, !0) : "undefined" != typeof XDomainRequest ? (t = new
XDomainRequest).open(e, n) : t = null, t } };

module.exports = Email;
```