Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт управления бизнес-процессами и экономики
Кафедра «Бизнес-информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.Н. Пупков
«____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03.02 «Прикладная информатика в менеджменте»

Разработка информационно-аналитической системы формирования отчетности на предприятии (на примере ООО «СпецПромАвтоматика»)

Руководитель _____ Доцент кафедры «БИ» А.В. Чубаров
подпись, дата

Консультант _____ Доцент кафедры «ЭУБП» Г.Ф. Яричина
подпись, дата

Выпускник _____ В.Е. Мартынов
подпись, дата

Нормоконтролер _____ Д.В.Спиридонов
подпись, дата

Красноярск 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Основополагающим условием успешного предприятия является наличие отчетности, которая может предоставить однозначную информацию о состоянии предприятия и может помощь в принятии управленческих решений. Эффективное формирование отчетности позволяет сократить время обнаружения проблемы на предприятии, а также способствуют получению большей прибыли и понимания состояния фирмы для руководителей и специалистов.

Возможности современных информационных технологий позволяют оптимизировать работу и деятельность на предприятии, а также более эффективно осуществлять его контроль и управление.

Информационно – аналитические системы – это современный высокоэффективный инструмент поддержки принятия тактических, стратегических и оперативных управленческих решений на основе оперативного и наглядного предоставления всей необходимой совокупности данных пользователям, ответственным за анализ состояния дел и принятие управленческих решений [1].

Объектом исследования выпускной квалификационной работы является ООО «СпецПромАвтоматика».

Предметом исследования являются бизнес – процессы отдела информационной аналитики в ООО «СпецПромАвтоматика».

Цель работы – провести анализ бизнес – процессов ООО «СпецПромАвтоматика», выявить проблемы в отделе информационной аналитики, на основе выявленной ситуации разработать и внедрить информационно – аналитическую систему формирования отчетности, которая повысит эффективность отдела.

Для реализации поставленной цели были определены следующие задачи:

- исследовать рынок систем спутникового мониторинга транспорта в России;
- провести анализ основных проблем, возникающих у предприятий, использующих данные системы, и рассмотреть возможные решения;
- исследовать деятельность и структуру ООО «СпецПромАвтоматика»;
- описать бизнес – процессы отдела информационной аналитики;
- выявить проблемные зоны в отделе информационной аналитики;
- усовершенствовать существующие бизнес – процессы;
- разработать информационно – аналитическую систему, позволяющую более эффективно формировать отчетность и оптимизировать использование рабочего времени сотрудников;
- внедрить информационно – аналитическую систему формирования отчетности;
- рассчитать экономическую эффективность от внедрения изменений.

При написании работы были использованы следующие ресурсы:

- метод проведения функционально – стоимостного анализа;
- сравнительный метод;
- измерительный метод;
- прогнозирование;
- метод обобщения;
- метод описания бизнес – процессов по методологии BPMN;
- метод составления экспертных оценок;
- методология разработки программных продуктов.

1 Обзор рынка систем спутникового мониторинга в России

1.1 Описание технологии систем спутникового мониторинга

Спутниковый мониторинг транспорта – система мониторинга подвижных объектов, построенная на основе системы спутниковой навигации, оборудования и технологий сотовой и/или радиосвязи, вычислительной техники и цифровых карт. Спутниковый мониторинг транспорта используется для решения задач транспортной логистики в системах управления перевозками и автоматизированных системах управления автопарком.

Принцип работы заключается в отслеживании и анализе пространственных и временных координат транспортного средства. Существует два варианта мониторинга:

- online, с дистанционной передачей координатной информации;
- offline, информация считывается по прибытии на диспетчерский пункт.

На транспортном средстве (ТС) устанавливается мобильный модуль, состоящий из следующих частей: приёмник спутниковых сигналов, модули хранения и передачи координатных данных. Программное обеспечение мобильного модуля получает координатные данные от приёмника сигналов, записывает их в модуль хранения и по возможности передаёт посредством модуля передачи.

Модуль передачи позволяет передавать данные, используя беспроводные сети операторов мобильной связи. Полученные данные анализируются и выдаются диспетчеру в текстовом виде или с использованием картографической информации.

В offline варианте необходимость дистанционной передачи данных отсутствует. Это позволяет использовать более дешёвые мобильные модули и отказаться от услуг операторов мобильной связи.

Мобильный модуль может быть построен на основе приёмников спутникового сигнала, работающих в стандартах NAVSTAR GPS или ГЛОНАСС.

GPS – спутниковая система навигации, обеспечивающая измерение расстояния, времени и определяющая местоположение во всемирной системе координат WGS 84. Позволяет почти при любой погоде определять местоположение в любом месте Земли (исключая приполярные области) и околоземного космического пространства. Система разработана, реализована и эксплуатируется Министерством обороны США, при этом в настоящее время доступна для использования для гражданских целей — нужен только навигатор или другой аппарат (например, смартфон) с GPS – приёмником [2].

Идея создания спутниковой навигации родилась ещё в 1950е годы. В тот момент, когда в СССР был запущен первый искусственный спутник Земли, американские учёные во главе с Ричардом Кершнером наблюдали сигнал, исходящий от советского спутника и обнаружили, что благодаря эффекту Доплера частота принимаемого сигнала увеличивается при приближении

спутника и уменьшается при его отдалении. Суть открытия заключалась в том, что если точно знать свои координаты на Земле, то становится возможным измерить положение и скорость спутника, и наоборот, точно зная положение спутника, можно определить собственную скорость и координаты [3].

GPS состоит из трёх основных сегментов: космического, управляющего и пользовательского. Схема реализации системы спутникового мониторинга (ССМ) представлена на рисунке 1.

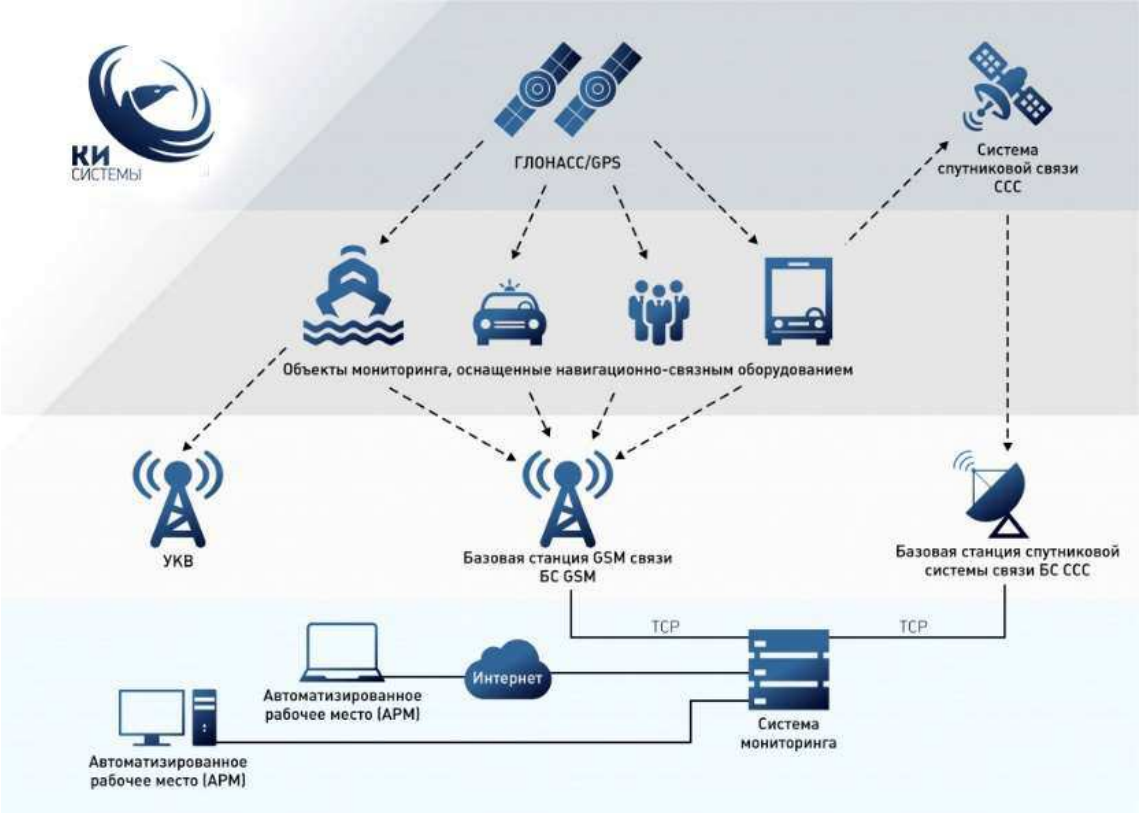


Рисунок 1 – схема реализации системы спутникового мониторинга

В общем виде принцип работы систем следующий – на транспорт устанавливается специальное бортовое навигационное оборудование мониторинга транспорта, в которое встроен ГЛОНАСС / GPS приемник, обрабатывающий сигналы со спутниковых навигационных систем. Эти устройства обычно называют абонентскими терминалами. Сейчас производители делают их многосистемными, чтобы увеличить качество приема сигнала. Так же в терминалы устанавливается SIM – карта сотового оператора, с помощью которой происходит передача данных в диспетчерский центр. Терминал получает навигационные сигналы, собирает телеметрические данные с дополнительных и штатных устройств и далее передает их на телематический сервер через коммуникационную среду. На сервере вся информация обрабатывается и далее отправляется диспетчеру или оперативному дежурному на автоматизированное рабочее место.

В зависимости от применяемых технических решений можно выделить пять поколений систем спутникового мониторинга транспорта:

Самые первые системы были оффлайновыми, то есть не позволяли осуществлять мониторинг в реальном времени. GPS – трекер записывал все данные в память и передавал их на сервер по прибытии транспортного средства на базу через проводной или беспроводной интерфейс. Такая схема позволяла контролировать маршрут автомобиля только постфактум и не способна помочь, например, при угоне автомобиля.

Во втором поколении для организации связи между GPS – терминалами и сервером использовались SMS либо механизм CSD. На сервер устанавливались один или несколько модулей сотовой связи, позволяющие принимать SMS или звонки с данными. Подобные системы отличались большим периодом времени между передачами данных местоположения и режимами получения данных по запросу. С массовым распространением мобильного интернета системы второго поколения практически вымерли.

В третьем поколении в качестве транспортной сети используются GPRS или EV – DO, что позволяет снизить расходы на передачу данных местоположения и строить системы отображения всех объектов в режиме реального времени. В таких системах сервер устанавливается непосредственно у клиента в локальной сети офиса, что обеспечивает лучшую оперативность и защищенность данных, однако требует регулярной поддержки сервера силами клиента. Обслуживание сервера требует определенной квалификации обслуживающего персонала на стороне клиента. На рабочие места пользователей устанавливается специализированное программное обеспечение. В некоторых системах допускается аренда ресурсов сервера, предоставляемых поставщиком услуг мониторинга.

Системы четвёртого поколения также используют один из механизмов мобильного интернет в качестве транспортной системы, но отличаются от третьего централизацией серверного обеспечения у поставщика услуги и использованием web – технологий. В этом случае сервер размещается у компании – поставщика, его мощности делятся между многими клиентами, а защищённый доступ к данным осуществляется через веб – приложение с любого компьютера, подключённого к интернету. Так как один сервер способен работать одновременно с тысячами объектов, значительно снижается стоимость внедрения и обслуживания системы. Одновременно может быть обеспечена более высокая надёжность хранения данных, так как компании – операторы способны построить сервер на базе качественного оборудования с многократным резервированием, содержать штат технических специалистов для круглосуточного обслуживания. Недостатком систем четвёртого поколения является полная централизация. Хотя вероятность аппаратного сбоя или наступления форс – мажорных обстоятельств в таких системах крайне низка, зато последствия сбоя могут стать весьма дорогостоящими и клиенту сложно оценить последствия утечки информации через технические службы оператора.

Системы мониторинга пятого поколения представляют собой глобальное развитие и централизацию систем предыдущего поколения в логически единый, распределённый центр мониторинга, работающий по принципу облачных технологий. В таком варианте данные GPS и ГЛОНАСС устройств, собираемые

коммуникационными серверами, стекаются в логически объединённый сервер базы данных и далее распределяются между промежуточными серверами, которые обеспечивают взаимодействие с пользователем. При такой архитектуре системы пользователи из разных регионов, стран и даже континентов получают информацию от ближайшего регионального центра с минимальной задержкой, получая от оператора программное обеспечение как услугу (англ. software as a service, сокр. SaaS). Некоторые платформы для спутникового мониторинга транспорта и управления им позволяют не только использовать стандартный интерфейс, но и персонализировать рабочее место под себя, тем самым, благодаря концепции облачных вычислений клиент получает рабочие места как услугу. Внедрение подобных систем даёт возможность глобального управления транспортными потоками в реальном времени, а пользователи могут экономить время, ресурсы и оптимально планировать маршруты.

По состоянию на 31 мая 2019 года исправно функционирует 31 космический аппарат [4].

Управляющий сегмент представляет собой главную управляющую станцию и несколько дополнительных станций, а также наземные антенны и станции мониторинга, ресурсы некоторых из упомянутых являются общими с другими проектами.

Пользовательский сегмент представлен приёмниками GPS, находящихся в ведении государственных институтов, и сотнями миллионов приёмных устройств, владельцами которых являются обычные пользователи.

Глобальная навигационная спутниковая система (ГЛОНАСС) – российская спутниковая система навигации. Система ГЛОНАСС, имевшая изначально военное предназначение, была запущена одновременно с системой предупреждения о ракетном нападении (СПРН) в 1982 году для оперативного навигационно–временного обеспечения неограниченного числа пользователей наземного, морского, воздушного и космического базирования, например, пассивных метео–РЛС типа РАЗК «Положение – 2». Дополнительно система транслирует гражданские сигналы, доступные в любой точке земного шара, предоставляя навигационные услуги на безвозмездной основе и без ограничений.

Основой системы являются 24 спутника, движущихся над поверхностью Земли в трёх орбитальных плоскостях [5]. Принцип измерения аналогичен американской системе навигации NAVSTAR GPS.

В настоящее время в России активно продвигается и лоббируется использование сигналов спутников ГЛОНАСС, разработка и производство клиентского оборудования мониторинга для этой системы. Принят ряд законодательных актов, которые форсируют внедрение ГЛОНАСС и ограничивают применение других систем [6].

При этом, в сравнении с NAVSTAR GPS, система ГЛОНАСС пока работает менее надёжно и в совокупности с наземным оборудованием даёт большую погрешность вычисления местоположения абонента [7]. Клиентское оборудование ГЛОНАСС стоит дороже, имеет большие размеры и худшие

параметры энергопотребления, представлено на рынке не так широко, как GPS [8].

На данный момент на рынке имеются системы, использующие оба стандарта навигации в процессе эксплуатации, чтобы компенсировать недостатки отечественной системы.

Системы спутникового мониторинга транспорта решают следующие задачи:

− мониторинг включает определение координат местоположения транспортного средства, его направления, скорости движения и других параметров: расход топлива, температура в рефрижераторе и др. Системы спутникового мониторинга транспорта помогают водителю в навигации при передвижении в незнакомых районах;

− контроль соблюдения графика движения — учёт передвижения транспортных средств, автоматический учёт доставки грузов в заданные точки и другие;

− сбор статистки и оптимизация маршрутов — анализ пройденных маршрутов, скоростного режима, расхода топлива и др. транспортных средств с целью определения лучших маршрутов;

− обеспечение безопасности — возможность определения местоположения помогает обнаружить угнанный автомобиль. В случае аварии система спутникового мониторинга помогает передать сигнал о бедствии в службы спасения. Также на основе спутникового мониторинга транспорта действуют некоторые системы автосигнализации.

Система спутникового мониторинга транспорта включает следующие компоненты:

− транспортное средство, оборудованное GPS или ГЛОНАСС контроллером или трекером, который получает данные от спутников и передаёт их на серверный центр мониторинга посредством GSM, CDMA или реже спутниковой и УКВ связи. Последние два актуальны для мониторинга в местах, где отсутствует полноценное GSM–покрытие, таких как Сибирь или Дальний Восток [9];

− серверный центр с программным обеспечением для приёма, хранения, обработки и анализа данных;

− компьютер диспетчера, ведущего мониторинг автомобилей.

Большинство GPS/ГЛОНАСС контроллеров и трекеров имеют схожие функциональные возможности:

− вычислять собственное местоположение, скорость и направление движения на основании сигналов спутников систем глобального позиционирования GPS или ГЛОНАСС;

− подключать внешние датчики через аналоговые или цифровые входы;

− считывать данные с бортового оборудования, имеющего последовательный порт или более специализированный интерфейс CAN;

− хранить некоторый объём данных во внутренней памяти на период отсутствия связи;

− передавать полученные данные на серверный центр, где происходит их обработка.

Ранее по причине слабого охвата территорий сетями мобильной связи GSM/3G широко использовались контроллеры, которые накапливали данные во внутренней памяти. По возвращению объекта в место основной дислокации (автопарк), данные переносились на сервер по проводным каналам либо через Bluetooth или Wi–Fi. Многие из существующих GPS–трекеров и контроллеров имеют открытый протокол взаимодействия с сервером, а также позволяют выполнять настройку режимов работы при помощи SMS, CSD или при помощи GPRS соединения.

Для получения дополнительной информации на транспортное средство устанавливаются дополнительные датчики, подключаемые к GPS или ГЛОНАСС контроллеру, например:

− датчик расхода топлива;
− датчик нагрузки на оси ТС;
− датчик уровня топлива (ДУТ) в баке;
− датчик температуры в рефрижераторе;
− датчики, фиксирующие факт работы или простоя специальных механизмов (поворот стрелы крана, работы бетоносмесителя), факт открывания двери или капота, факт наличия пассажира (такси).

Полученные данные могут либо накапливаться в локальном устройстве и затем переноситься в центральную базу по возвращении в парк, либо передаваться на центральный сервер в режиме реального времени, обычно по каналам сотовой связи.

Датчики и трекеры могут устанавливаться на транспортном средстве скрытым образом.

Системы спутникового мониторинга, представленные в России, можно условно разделить на несколько групп:

− трекеры с минимальным набором программного обеспечения, часто бесплатным, которое позволяет решать базовые задачи персонального мониторинга;
− программно–аппаратные комплексы, представляющие собой законченные решения. В этом случае спутниковое оборудование и программное обеспечение встроены друг на друга, и переход с одной на другую систему затруднён;
− программные комплексы, совместимые с различными контроллерами и трекерами, предоставляемые в аренду с серверных центров в формате Software as a service;
− программные комплексы для серверной установки, способные поддерживать различные виды GPS и ГЛОНАСС оборудования одновременно, позволяющие клиентам иметь различные контроллеры в своём автопарке;
− комплексные услуги по мониторингу автомобилей, которые оказываются специализированными компаниями. В таком случае клиент платит ежемесячную абонентскую плату за использование системы. Отдельно оплачивается приобретение и установка контроллеров на транспортные

средства, при этом некоторые компании предлагают аренду контроллеров, тем самым снижая единовременные затраты для компании, которая планирует вести мониторинг своего автопарка.

Следует также учитывать, что системы мониторинга могут быть как самостоятельными решениями, так и модулем в более сложной системе TMS и/или FMS. Немаловажное значение имеют возможности выполнения системой бухгалтерской, складской, логистической функций или интеграции системы спутникового мониторинга с другими автоматизированными системами управления предприятием.

На российском рынке услуга GPS/ГЛОНАСС–мониторинга автотранспорта существует достаточно давно. Рынок систем GPS/ГЛОНАСС–наблюдения за транспортом начал стремительно развиваться в 2005 году и ежегодно прирастает [10]. Это связано, в первую очередь, с тем, что системы мониторинга автотранспорта приносят видимый экономический эффект и сроки окупаемости системы достаточно коротки.

Основными мировыми рынками, активно применяющими системы спутникового мониторинга автотранспорта, являются рынки таких стран, как Россия, Европа, Северная и Латинская Америка.

В России рынок спутниковых систем мониторинга представлен большим количеством компаний, занимающихся разработкой навигационного оборудования. Основные компании, производящие системы спутникового мониторинга в России [11]:

- Galileosky — российская компания занимается производством оборудования для спутникового мониторинга: GPS/ГЛОНАСС терминалы, которые имеют функционал, начиная от базового до специфичного. Кроме того, специалисты Galileosky создали уникальную технологию Easy Logic для самостоятельного программирования функционала терминалов, с помощью которой система мониторинга может задать для различных терминалов свои алгоритмы действий при возникновении того или иного события. Разработка внедрена только в терминалы компании.;
- Teltonika — это международная литовско–финская компания, предлагающая своим клиентам линейку GPS–трекеров для мониторинга транспорта, а также для персонального мониторинга;
- Arusnavi — это российский производитель устройств спутникового мониторинга: навигационные контролеры, датчики уровня топлива, платы расширений. Специализируются на разработке оборудования для персонального мониторинга и мониторинга движущихся объектов, т.е. автомототранспорт, велосипеды;
- TechnoKom — созданная в 1993 году, группа компаний «ТехноКом» удерживает лидирующие позиции на рынке России в области разработки и производства систем ГЛОНАСС/GPS спутникового мониторинга транспорта, персонала, датчиков контроля топлива и программного обеспечения [12];
- Neomatica — российский разработчик и производитель оборудования ADM для систем ГЛОНАСС/GPS мониторинга. Инновационные

решения компании направлены на создание эффективной системы по автоматизации контроля транспорта и оптимизации различных бизнес–процессов [13];

− АРК СОМ — компания АПК КОМ – российский разработчик и производитель систем спутникового мониторинга. Более сотни тысяч устройств и внедрены и работают в 73 регионах России, на Украине, в Польше и Казахстане [14];

− Navtelecom — компания ООО «Навтелеком» образована в 2007 году. В компании основные усилия приложены на разработку и производство оборудования для ГЛОНАСС/GPS мониторинга транспорта, GSM–охраны недвижимости и беспроводного видеонаблюдения;

− Satellite Solutions — является одним ведущих разработчиков и производителей навигационного (ГЛОНАСС/GPS) оборудования в России и странах СНГ. Решения «Satellite Solutions» позволяют решать любые задачи в сфере гражданского мониторинга;

− OMNICOMM — разработчик и производитель системы мониторинга транспорта Omnicomm Online, датчиков уровня топлива Omnicomm LLS и сопутствующего оборудования.

Для большего понимания представленных систем и потребности в данных продуктах рассмотрим объем рынка на 2018 год:

− По итогам 2018 года объем основных сегментов рынка составил около 14,9 млрд руб. и вырос на 8,5% по сравнению с данными 2017 года (13,7 млрд руб.);

− Структура рынка: 10,5 млрд руб. приходится на доходы от абонентской базы платформ для мониторинга транспорта, по 2,1 млрд руб. – на сегменты терминалов и датчиков уровня топлива;

− Более 2 млн. транспортных средств и стационарных объектов подключено к системам мониторинга транспорта, объем накопленной абонентской базы вырос на 6,6% по сравнению с 2017 годом (1,95 млн. объектов);

− Прирост абонентской базы у основных разработчиков ПО для мониторинга транспорта составил: OMNICOMM – 27%, Gurtam – 14%, ТехноКом – 7%.

Согласно данным «Автостат», на начало 2019 года доля грузового и легкого коммерческого транспорта в структуре российского автопарка, который преимущественно использует телематику, составила 14,4% или около 8,3 млн. транспортных средств. Эксперты аналитического центра Omnicomm отмечают, что в 2018 году рынок мониторинга транспорта продемонстрировал стабильный рост – в пределах 8%. Объем новых подключений составил 250 000 терминалов и 190 000 датчиков уровня топлива. По оценке Omnicomm, суммарная доля проникновения технологий мониторинга транспорта достигла 15–17%, но этот показатель существенно отличается по сегментам. Самый высокий показатель – до 50% – в дальнорейсовых перевозках, а в легком коммерческом транспорте доля составляет всего 5%.

Доля доходов от абонентской базы ПО составляет более 70% рынка мониторинга транспорта. Лидерами по приросту абонентской базы стали российская платформа Omnicomm Online/Omnicomm (+27% по сравнению с 2017 годом), Wialon/Gurtam (+14%) и АвтоГраф/ТехноКом (+7%). В сегменте терминалов для мониторинга транспорта (15% рынка) лидерами являются OMNICOMM (25%), Навтелеком (19%), который предлагает оборудование в бюджетном ценовом сегменте, и ТехноКом (17%). В сегменте датчиков топлива (также 15%) лидирующие позиции удерживает ONMICOMM (37%), ТехноКом (16%) и «Эскорт» (16%).

На рисунке 2 представлена средняя стоимость для клиента на использование одного терминала ССМ.

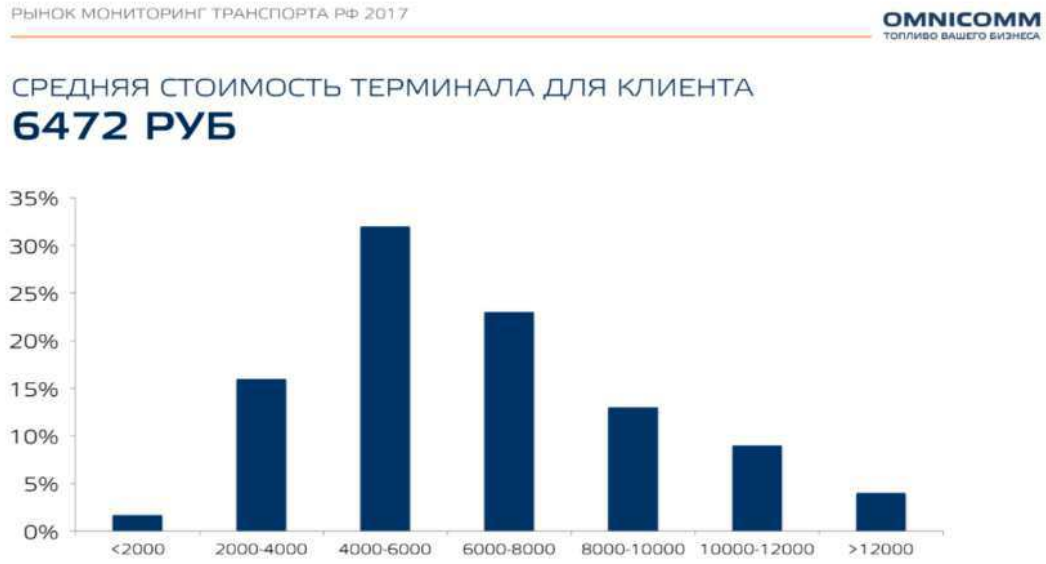


Рисунок 2 – средняя стоимость терминала ССМ для клиента

На рисунке 3 указана средняя стоимость абонентской платы для клиента.

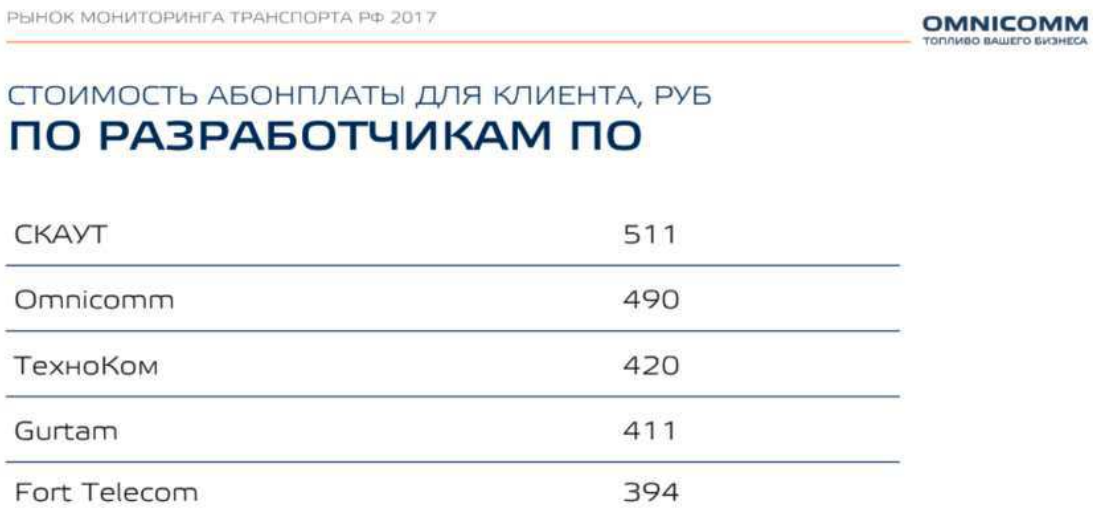


Рисунок 3 – средняя стоимость абонентской платы для клиента

Количество дилеров представлено на рисунке 4.

ПОРТРЕТ ДИЛЕРА/ИНТЕГРАТОРА

OMNICOMM
ТОПЛИВО ВАШЕГО БИЗНЕСА

	Количество интеграторов, шт	Количество «дилеров», шт
Gurtam	517	391
ТехноКом	182	111
Omnicomm	160	109
СКАУТ	95	54

Рисунок 4 – Количество дилеров основных производителей систем спутникового мониторинга

Средняя стоимость датчика уровня топлива (ДУТ) показана на рисунке 5.

РЫНОК ДАТЧИКОВ УРОВНЯ ТОПЛИВА РФ 2017

OMNICOMM
ТОПЛИВО ВАШЕГО БИЗНЕСА

СТОИМОСТЬ ДУТ ДЛЯ КЛИЕНТА, РУБ
ПО ПРОИЗВОДИТЕЛЯМ

Omnicomm	8 734
СКАУТ	8 544
ТехноКом	7 674
Эскорт	6 862
Италон	6 477

Рисунок 5 – средняя стоимость ДУТ по производителям.

Распределение рынка по главным производителям указано на рисунке 6.

Рисунок 6 – Распределение рынка по производителям систем

Как показывает практика, цена на приборы спутникового мониторинга автотранспорта эластична и может меняться как исходя из спроса на рынке, так и в зависимости от предложения. Чем выше конкуренция, тем ниже цена. Это продиктовано тем, что под воздействием конкуренции каждая компания стремится предложить более выгодные условия сотрудничества своим потенциальным клиентам, предлагая различные скидки. Максимальные стоимости систем спутникового мониторинга автотранспорта определяются условиями эксплуатации (степень защищенности от тех или иных воздействий: механические, электромагнитные и так далее) и функциональным назначением (количество подключаемых датчиков, возможность подключения к CAN–шине (бортовому компьютеру), наличие «черных ящиков» и так далее).

Помимо профильных участников рынка, свои услуги по спутниковому мониторингу автотранспорта предлагают ведущие операторы сотовой связи.

Например, в середине 2013 г. компания «Мегафон» начала продавать услугу «Контроль автопарка» для корпоративных клиентов, которая позволяет в режиме реального времени контролировать параметры движения автотранспорта и отслеживать показания различных датчиков, установленных на нем. По словам представителя компании, ежемесячно количество пользователей услуги растет на 10–15%».

Относительно данных, показанных ранее, можно с уверенностью сказать, что данный рынок в России достигает своего пика.

Большинство компаний, производящие системы спутникового мониторинга работают через дилерскую сеть. В Красноярском крае основные компании, занимающиеся ССМ:

- STAVTRACK;
- Краевой Центр Коммуникации;
- ГК Нави;

– M2M;
– ПК СКТ;
– Импульс;
– ГК «ГУГОЛ»;
– ГК «Антей».

Данные компании имеют дилерские договора с основными производителями терминалов и датчиков. С данными договорами они выполняют монтажные, ремонтные работы, консультационные услуги, настраивают ПО под нужды организаций.

Экономический эффект от внедрения систем мониторинга и управления подвижными объектами, с учетом специфики парка оборудуемого автотранспорта и характера перевозимых грузов, выражается в следующих показателях:

− минимизация потерь от краж груза, угонов транспортного средства и его нецелевого использования;

− минимизация затрат на техническое обслуживание и горюче–смазочных материалы за счет оптимизации маршрутов и снижения непродуктивного пробега автотранспорта;

− снижение потребности в расширении парка автотранспорта;

− повышение транспортного обслуживания клиентов и возможность привлечения новых клиентов за счет расширения спектра услуг и оперативного реагирования на запросы;

− оптимизация планирования работы на основе объективной информации о реальном пробеге автотранспорта и снижение потерь, связанных с его ремонтом и простоем;

− повышение эффективности работы персонала и возможность введения системы материального стимулирования, базирующейся на достоверной информации о работе каждого водителя и поощряющей более эффективное использование рабочего времени, транспорта, горюче–смазочных материалов и специального оборудования и так далее.

Использование систем мониторинга и управления подвижными объектами в повседневной деятельности компаний позволяет по различным оценкам снизить пробег автотранспорта и топливные расходы на 15–30%, увеличить объем предоставляемых услуг на 25%, повысить дисциплину водителей и производительность труда на 30%, сократить расходы на ремонт на 10%.

1.2 Обзор программных продуктов, использующихся в системах спутникового мониторинга

Самым существенным различием многих систем спутникового мониторинга, представленных на рынке, является функциональность серверного и клиентского программного обеспечения, возможность разносторонне обрабатывать данные, генерировать отчёты.

Функции серверного центра может выполнять как обычный компьютер с установленным программным обеспечением для простых систем мониторинга, так и распределённая серверная система с использованием нескольких серверов, выполняющих разные задачи, способная вести одновременный мониторинг десятков тысяч автомобилей и обеспечивать подключение к серверному центру нескольких тысяч пользователей (диспетчеров) одновременно.

Диспетчерское программное обеспечение для спутникового мониторинга автомобилей можно условно разделить на несколько типов:

– ПО, содержащее все компоненты, включая карты и базу данных движения объектов на единственном компьютере;

– ПО, имеющее клиентскую часть, которая устанавливается на компьютеры диспетчеров;

– ПО, использующее web–интерфейс, что позволяет избежать установки каких–либо специальных компонентов и вести мониторинг с любого компьютера, подключённого к Интернет.

Разновидностью последнего варианта является ПО, использующее трёхуровневую архитектуру, когда компоненты и функции центра обработки данных распределены между несколькими серверами: базы данных, картографической подсистемы, телекоммуникационным сервером и сервером приложения, обеспечивающего работу web–интерфейса пользователя.

Трёхуровневая архитектура (трёхзвенная архитектура, англ. three–tier) — архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений) (Рисунок 7).

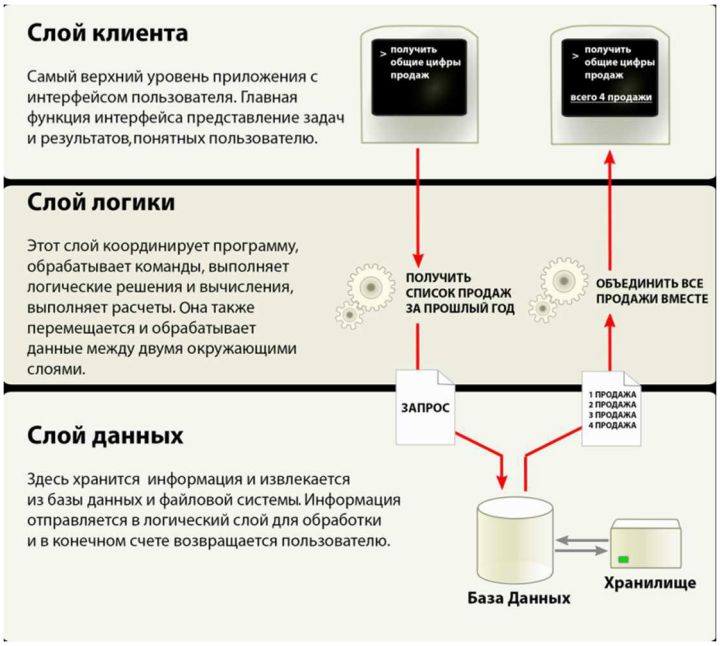


Рисунок 7 – Трехуровневая архитектура

В то время, как первый и второй типы систем остаются надёжным решением для специальных применений, где использование каналов Интернет невозможно из–за низкого качества последней мили или запрещено нормативными актами, последний тип систем имеет ряд преимуществ и позволяет компаниям–операторам увеличить охват рынка, ускорить внедрение мониторинга, переводя его в разряд платной услуги. На специализированной выставке «НАВИТЕХ» 2015 года [15] web–системы были представлены от компаний 1С (1С:Центр спутникового мониторинга (компания–разработчик ITOB) , M2M телематика (Россия), Gurtam (Белоруссия) и ГК СКАУТ (Россия), клиентское ПО представляли компании Level (Чехия) и ОАО «Русские Навигационные Технологии», ООО «ТехноКом», ЕНДС, система ТрансКонтроль и M2M телематика. Большинство производителей современных систем мониторинга включают в свои продукты возможность работы диспетчеров через web–интерфейс и построения распределённых систем серверов.

Программное обеспечение для спутникового мониторинга обычно имеет ряд интерфейсов. Вход пользователей в систему мониторинга чаще всего защищён паролем для предотвращения несанкционированного доступа к информации. В системах существует определённая иерархическая структура, при которой администратор системы мониторинга управляет правами доступа различных пользователей к различным объектам мониторинга и различным функциям программы.

Самые распространённые функции, которые присутствуют в большинстве систем спутникового мониторинга:

- подключение и настройка трекеров в системе;
- подключение и настройка датчиков в системе;
- мониторинг текущего положения транспорта на карте;
- мониторинг состояния приборов и датчиков транспортного средства;
- просмотр маршрута перемещения и пробега автомобиля за выбранный интервал времени;
- создание точек интереса и геозон на карте;
- контроль перемещения из/в геозоны;
- настройка уведомлений, высылаемых системой, когда происходят определённые события (превышение скорости, слив топлива и др.);
- настройка шаблонов отчётов, выполнение отчётов;
- построение графиков на основании данных системы;
- управление объектами мониторинга через SMS команды или CSD соединение;
- создание маршрутов и путевых точек, контроль соблюдения маршрута.

Геозона— виртуальный произвольно ограниченный участок на географической карте. Геозоны используются в системах спутникового мониторинга для задания виртуального периметра, при пересечении границ

которого происходит оповещение пользователя или выполняются различные команды [16].

Дополнительные функции, которые расширяют возможности системы спутникового мониторинга:

- поиск ближайшего к заданной точке автомобиля;
- передачу текстовых сообщений водителю транспортного средства и обратно, от водителя к диспетчеру;
- обеспечение голосовой связи с водителем;
- ведение журнала техобслуживания автомобиля;
- определение периметра и площади объектов на карте;
- web–доступ в систему мониторинга с мобильного телефона или КПК;
- экспорт из отчётов в форматы, поддерживаемые иным ПО (Excel, Pdf, XML, CSV и др.);
- изменение иконок, отображающих объекты на карте;
- передача данных от другого оборудования, установленного на транспортном средстве (тахограф, датчик уровня топлива).

SaaS (англ. software as a service — программное обеспечение как услуга) — одна из форм облачных вычислений, модель обслуживания, при которой подписчикам предоставляется готовое прикладное программное обеспечение, полностью обслуживаемое провайдером. Поставщик в этой модели самостоятельно управляет приложением, предоставляя заказчикам доступ к функциям с клиентских устройств, как правило через мобильное приложение или web–браузер [17].

Каждый производитель собственных систем спутникового мониторинга предоставляет свое ПО для мониторинга, имеющая различный функционал. Рассмотрим основных участников рынка.

Компания «Техноком» разрабатывает «АвтоГРАФ». Перечень ПО, которое они предоставляют:

- диспетчерское ПО АвтоГРАФ 5 PRO;
- web–клиент AutoGRAPH Web (Рисунок 8);
- службу AutoGRAPH.NET Service, позволяющее интегрировать данные по объектам мониторинга в сторонние сервисы;
- серверное приложение AutoGRAPH Server, который позволяет принимать данные с модулей посредством передачи данных через GPRS или Ethernet;
- AGDataLoader – служба Windows, обеспечивающая ретрансляцию данных между серверами в IT–архитектуре предприятия;
- программа–конфигуратор контроллеров АвтоГРАФ – GSMConf.

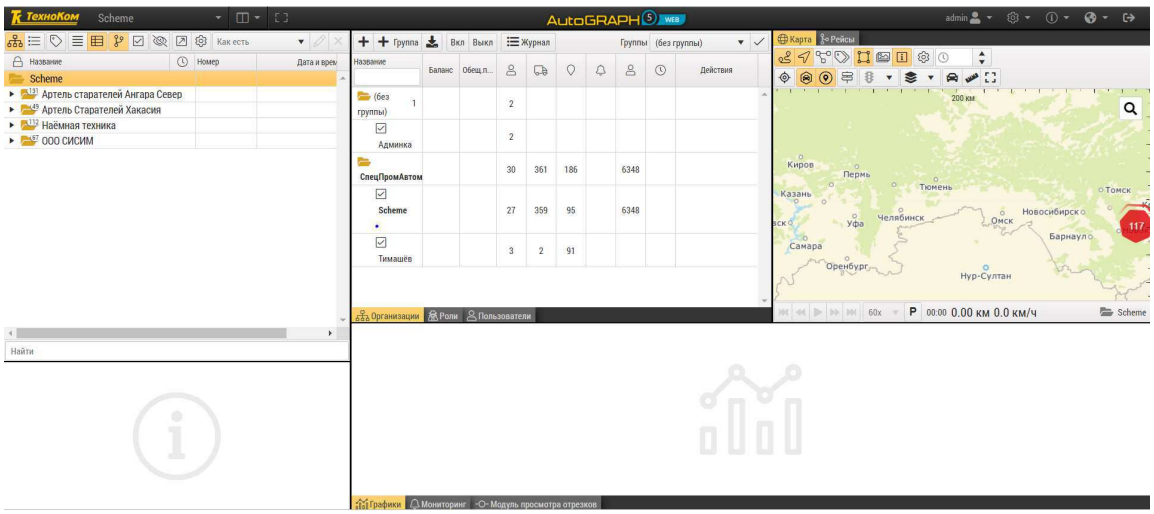


Рисунок 8 – интерфейс пользователя в AutoGRAPH Web

Основным преимуществом является бесплатное предоставление своего ПО компаниям, при условии использования их терминалами. Данные с других терминалов ПО напрямую не обрабатывает, однако есть возможность ретранслировать данные через дилеров.

Компания Omnicomm предоставляет Omnicomm online – web–приложение для мониторинга ТС (Рисунок 9).

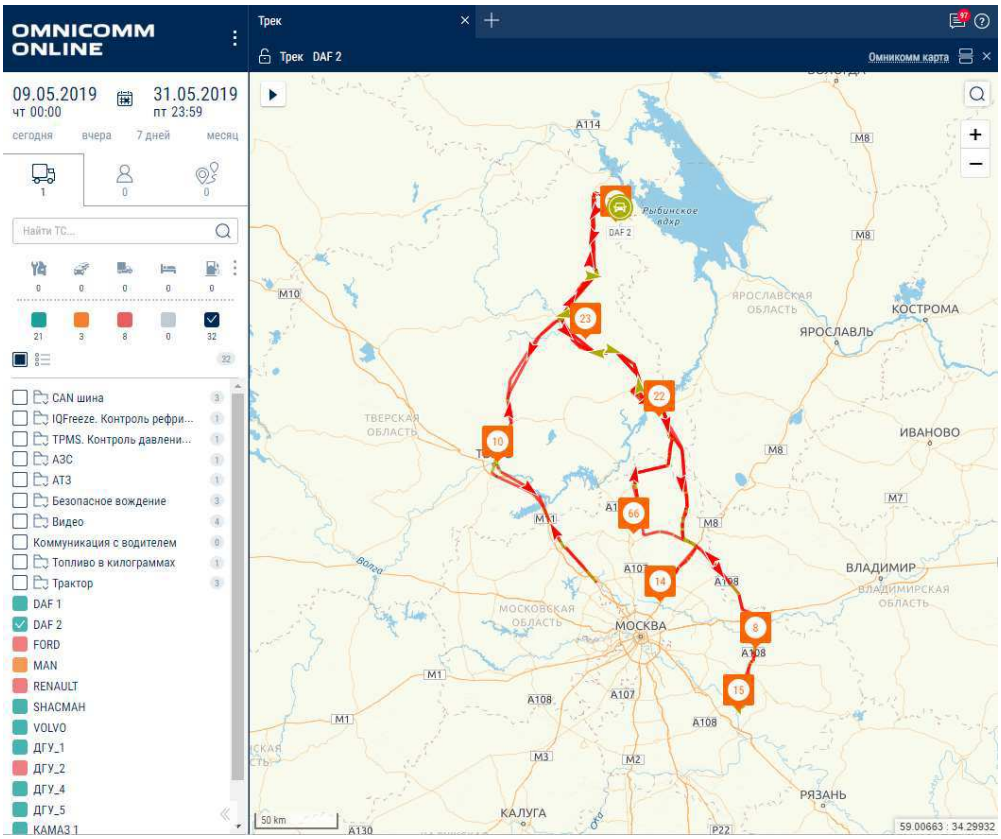


Рисунок 9 – Внешний вид Web–клиента Omnicomm online

ГК СКАУТ – Скаут–Платформа это программная часть Системы СКАУТ, которая обеспечивает доступ клиентов к данным о своих транспортных

средствах, их комплексную аналитику и интеграцию в корпоративную информационную систему заказчика. Включает в себя:

– СКАУТ – Online – WEB–интерфейс для руководителей и менеджеров. Предназначен для оперативного доступа к данным Системы СКАУТ с любого устройства – персональный компьютер, ноутбук, планшет или смартфон. Не требует никаких дополнительных установок и компонентов (Рисунок 10);

– Программа СКАУТ–Студио является рабочей программой диспетчера (оператора) программно–аппаратного комплекса СКАУТ. Предназначена для упрощения работы диспетчеров транспортных предприятий, сокращения расходов на содержание транспорта и повышения эффективности работы предприятий;

– СКАУТ–СильверСтудио – один из пользовательских интерфейсов СКАУТ–Платформы. Для работы «СильверСтудио» необходим интернет–обозреватель с установленным плагином Silverlight [18].

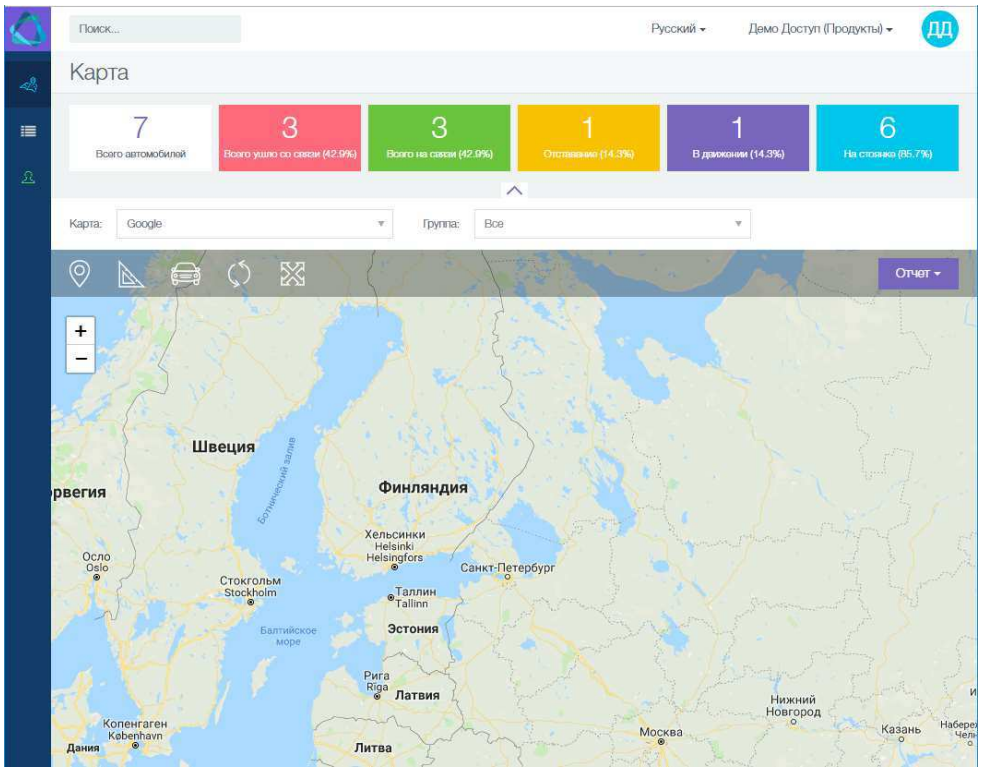


Рисунок 10 – Интерфейс «СКАУТ – Online»

Gurtam – Wialon – платформа для GPS/ГЛОНАСС мониторинга и IoT (Рисунок 11).

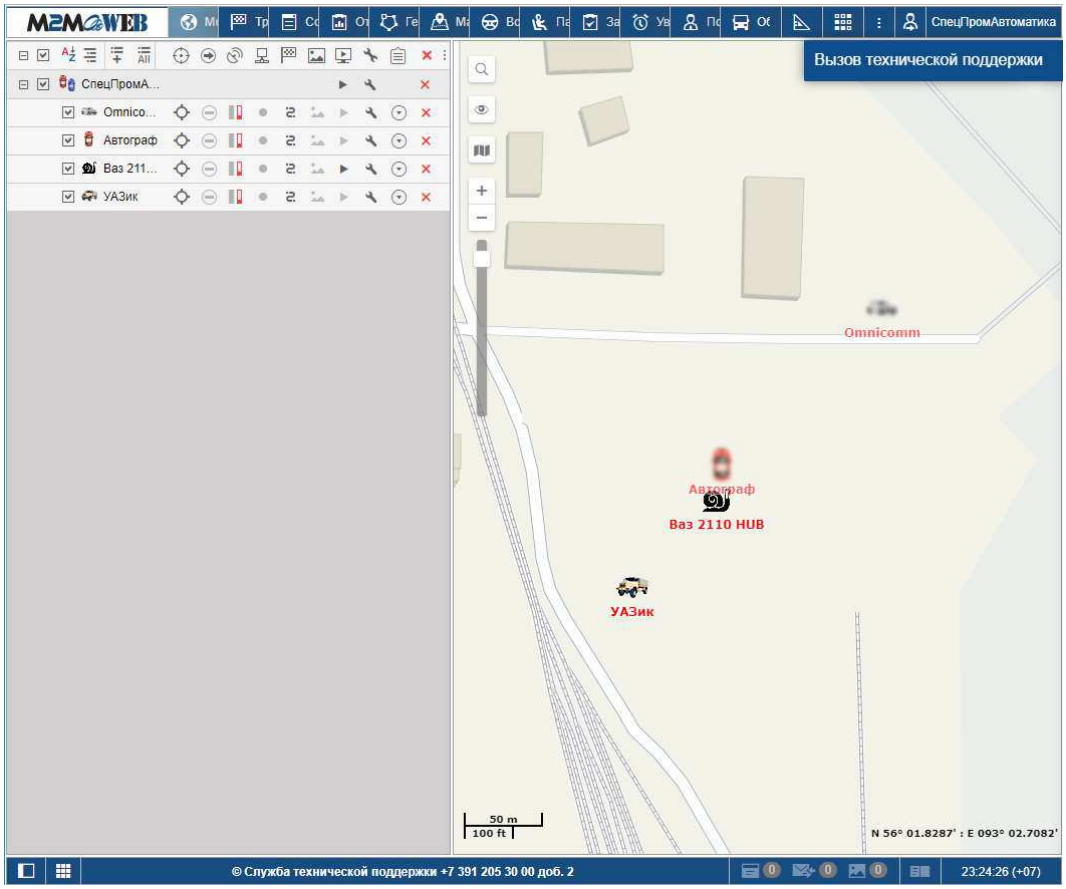


Рисунок 11 – интерфейс программы Wialon

В таблице 1 приведено более детальное сопоставление ПО систем спутникового мониторинга автотранспорта.

Таблица 1 – сравнение возможностей популярных программных средств на рынке систем спутникового мониторинга

Параметр	СКАУТ	АвтоГРАФ	Omnicomm	Wialon
Основные функции				
Принцип получения данных с сервера (репликации)	Оптимизированный режим репликации (раз в 5 сек.)	Загрузка данных с сервера "по запросу"	Загрузка данных с сервера "по запросу"	Загрузка данных с сервера "по запросу"
Требования к каналу подключения к серверу	Минимальные	Высокие	Высокие	Высокие
Загрузки данных по 1 автомобилю за 1 неделю	менее 1 сек.	около 1 мин	менее 3 сек.	менее 1 сек.
Возможность работы со старыми данными без подключения к серверу	Да	Нет	Да	Нет

Продолжение таблицы 1 – сравнение возможностей популярных программных средств на рынке систем спутникового мониторинга

Параметр	СКАУТ	АвтоГРАФ	Omnicomm	Wialon
Карты				
Векторные	ИНГИТ, City Guide, Резидент, КБ Панорама, польский (текстовый) формат	Свой закрытый формат и ИНГИТ	Ингит, Резидент	Ингит, Резидент
Растровые	Да	Нет	Нет	Нет
Возможность использования карт заказчика	Да	Нет	Нет	Нет
Отчеты				
Количество доступных отчетов	22	5	10	2
Просмотр построенных отчетов	в самой программе диспетчера	во внешних программах (EXCEL, AcrobatReader)	в самой программе диспетчера	в самой программе диспетчера
Интерактивность сложных отчетов	да. Статистика в сложном отчете может быть мгновенно "развернута" в простой отчет	Нет	Нет	Нет
Интерактивность простых отчетов	да. Любое событие в отчете можно мгновенно отобразить на карте (стоянка, заправка, слив, срабатывание датчика и др.)	Нет	Нет	Нет
Сохранение отчетов в форматы	Adobe PDF, Microsoft XPS, HTML, MHT Web, Text,Microsoft Word, Open Document Writer, Excel, Excel Xml, Excel 2007, Open Document Calc, CSV, DBF, XML, BMP, GIF, JPEG, PCX, PNG, TIFF	Excel, HTML	Excel, PDF, HTML	Excel, PDF
Время построения типового отчета	около 10 сек	около 20 сек	более 1 мин.	Более 1 мин.
Достаточность информации в стандартных отчетах	Да	Нет	Нет	Да

Окончание таблицы 1 – сравнение возможностей популярных программных средств на рынке систем спутникового мониторинга

Параметр	СКАУТ	АвтоГРАФ	Omnicomm	Wialon
Возможности контроля нарушений в режиме реального времени				
Наличие настраиваемой системы сообщений в случае нахождения транспорта в заданной зоне больше заданного порога	Да	Нет	Нет	Да
Наличие настраиваемой системы сообщений в случае остановки транспорта вне заданных зон	Да	Нет	Нет	Да
Возможность создания графических зон любой формы	Да	Да	Да	Да
Возможность формирования групп объектов и а/м, перенос между группами	Да	Да	Да	Да
Возможность наблюдения только выбранных транспортных средств с отображением зон	Да	Да	Да	Да

Резюмируя обзор программных продуктов можно сказать, что функционал позволяет удовлетворить большинство потребностей бизнеса.

1.3 Обзор проблем в отрасли систем спутникового мониторинга.

Как и везде, в отрасли систем спутникового мониторинга имеются различного рода трудности, связанные с технической частью, программным обеспечением и с точки зрения ведения и взаимодействия бизнеса. В первую очередь, рассмотрим проблемы систем спутникового мониторинга с точки зрения работы терминалов и различных датчиков. Основные проблемы:

- зависимость от зоны покрытия;
- отсутствие автономной работы на многих терминалах;
- физические повреждения датчиков и модулей.

В первую очередь, при потребности бизнеса в мониторинге транспорта необходимо учитывать специфику работы данных транспортных средств. Если ТС работают в городе или пригороде, проблем с передачей данных нет, так как город покрыт сотовой связью, и терминал будет постоянно на связи, успешно

ретранслировать данные. В случае, если машины работают в труднодоступных участках (золотодобывающие участки, лесопроизводство и другое), необходимо понимать, что в таких зонах может отсутствовать сотовая связь.

Чтобы минимизировать данный фактор, перед монтажом необходимо учесть зону покрытия местности, и при условии наличия связи какого–либо сотового оператора, покупать для ретрансляции данных SIM–карту данного оператора. Тем не менее, возможна ситуация, при которой сотовой связи на месте работы транспорта нет, однако получать данные необходимо. В таком случае на данном участке есть возможность развернуть локальную сеть с Wi–Fi точками, которые охватывают зону работы на участке. Для данного решения необходимо приобретать комбинированные модули (GSM+Wi–Fi), у многих производителей ССМ имеется данное решение (GalileoSky, АвтоГРАФ). Если нет возможности на участках работы мобильных единиц создать сеть из Wi–Fi точек и нет сотовой связи, можно оборудовать отдельное транспортное средство с модулем получения данных с приборов. Для минимизации расходов данный модуль можно разместить на топливозаправщике, который заправляет технику.

Следующая проблема заключается в отсутствии автономной работы многих терминалов при отсутствии питания от транспортного средства. При отсутствии питания по различным причинам (механическое повреждение кабеля питания, поломка транспортного средства) возможна потеря данных и поломка терминала, если произошел скачек напряжения или подобный прецедент.

Предлагаемым решением данной проблемы рассматривается подключение к модулю дополнительного автономного источника питания, дополнительного аккумулятора. Предотвращение механических повреждений кабеля питания можно осуществить регламентированным порядком монтажа оборудования и использованием кабелей защищённого типа.

Наконец, всегда имеется риск физического повреждения терминалов и различных типов датчиков вследствие неправильной эксплуатации транспортного средства или условий, в котором оно работает.

Решением данной проблемы является установка дополнительной защиты терминалов и датчиков на транспортное средство, предусмотренной производителем или собственными решениями. Также, для уменьшения риска поломки терминалы следует устанавливать в наиболее безопасных местах транспортного средства, которые различаются от типа техники и выполняемых ею работ.

Следующей основной проблемой являются компании, распространяющие терминалы спутникового мониторинга и датчики, которые работают с производителем по дилерскому договору. При данной реализации продукции и российских реалиях бизнеса компания–производитель далеко не всегда может корректно отслеживать деятельность компаний–дилеров, поэтому качество обслуживания, сопровождения и иных услуг для конечной компании, заинтересованной в подключении и обслуживании систем спутникового мониторинга, может быть недостаточной. Это сказывается как на качестве

монтажа, так и на последующем сопровождении оборудования и предоставление данных по транспортным средствам.

Данную проблему можно минимизировать крупным предприятиям, способным внутри своей организационной структуры создать отдел монтажа систем спутникового мониторинга, собственного сервисного центра по ремонту оборудования, подразделение поддержки программного обеспечения ССМ и отдел аналитики данных по транспорту. Мелким предприятиям, которые не обладают достаточными ресурсами для самостоятельного обслуживания спутникового мониторинга, необходимо тщательно выбирать компанию, которая будет этим заниматься, а также точно определить границы взаимодействия с дилером, которые будут четко описаны в договоре на обслуживание.

Кроме данных проблем, отдельно можно выделить проблему с программным обеспечением, цельностью данных, получаемых конечным пользователем, а также наличие не валидных данных с прибора.

С точки зрения программного обеспечения, всегда есть вероятность ошибки в программном продукте, которая может привести к ошибкам в системе и невозможности осуществлять мониторинг. Кроме того, в выбранном программном обеспечении могут отсутствовать функции, необходимые для предприятия. Наконец, не все компании, поставляющие и обслуживающие терминалы и/или программное обеспечение в достаточной мере проводят обучение и консультацию сотрудников конечного предприятия, вследствие чего многие возможности могут быть упущены по причине неосведомленности сотрудников. Результатом может являться упущенная экономическая выгода для предприятия.

Решение данной проблемы аналогично ранее описанному: создание внутреннего подразделения, ответственного за ПО ССМ, тем самым изучение всех составляющих системы. Также, по поводу ошибок в ПО разработчика систем спутникового мониторинга необходимо оперативно сообщать обнаруженный баг или недочет, чтобы ускорить выпуск обновления ПО, которое его исправит.

Также потери данных по транспортному средству при ретрансляции на основной сервер, данная ситуация в основном связана с модулями, которые передают данные через Wi–Fi сеть. Возникает ненулевая вероятность того, что транспортное средство может переслать данные за разные периоды времени через две и более Wi–Fi сети (техника перемещается между различными участками), которые отправляют данные на разные сервера. В итоге, при пересылке данных на главный сервер возникают ситуации когда часть данных теряется, из–за внутренних алгоритмов перезаписи, данные «затираются». В данном случае единственным решением проблемы является переработка сети ретрансляции данных для исключения ситуации записи данных одной транспортной единицы на разные промежуточные сервера.

Можно выделить ряд проблем для пользователей систем, связанных непосредственно с формированием отчетности:

- в полученных данных наблюдаются различного рода ошибки;

− проблема с корректным отображением «сырых» данных для конечного пользователя;

− вероятность пропустить инцидент с техникой при возникновении человеческого фактора.

Проблема некорректных данных по транспортной единице может возникать при неправильной работе модуля или датчиков, или эксплуатации датчиков и модулей в плохих условиях. Для дополнительной фильтрации таких данных в ПО предусмотрены различные инструменты, такие как усреднение значений по различным параметрам, исключения бросков координат и так далее.

Наконец, при большом парке есть вероятность пропустить событие с единицей техники, которое негативно скажется на деятельности предприятия (слив топлива, критическое превышение температуры двигателя, несоответствие давления в шинах и так далее), что, в свою очередь, также приведет к экономическим потерям и снижению эффективности мониторинга транспортных средств. В данном случае, чтобы максимально исключить подобные ситуации, на предприятии необходимо создать структурное подразделение, отвечающее за непосредственный анализ данных с транспортных средств и предоставляющую оперативные сводки по состоянию парка техники. Кроме того, необходимо равномерно распределять нагрузку между сотрудниками данного подразделения, то есть учитывать объем работы по каждой единице, количество машин, оборудованных системами спутникового мониторинга, а также количество созданной отчетности в данном отделе. В ином случае, при чрезмерной нагрузке на специалистов, появляется вероятность, во–первых, также пропустить какой–нибудь инцидент с транспортным средством, а во–вторых, снизить оперативность предоставления информации за отчетный период.

Таким образом видно, что в данной отрасли имеются проблемы разного характера, с некоторыми из которых необходимо справляться самим разработчикам и производителям, а решение других проблем возможно только с участием компаний–пользователей системы спутникового мониторинга.

2 Анализ предприятия ООО «СпецПромАвтоматика»

2.1 Анализ организационной структуры и деятельности предприятия

Компания ООО «СпецПромАвтоматика» является дочерней фирмой золотодобывающего холдинга «Сибзолото», который, в свою очередь, объединяет четыре артели:

- ООО «Артель старателей Ангара–Север»;
- ООО «Артель старателей Хакассия»;
- ООО «Сисим»;
- ООО «Артель старателей Июсская».

Предприятие было зарегистрировано и действует с 11.12.2013 года, соответственно, ему более 5 лет.

Компания ООО «СпецПромАвтоматика» выросла на базе крупнейшего в Красноярском крае золотодобывающего холдинга. У компании следующие специализации:

- монтаж систем спутникового мониторинга на транспортные средства – холдинга и подрядчиков;
- ремонт и проведение ревизии уже установленных систем спутникового мониторинга;
- проведение электромонтажных работ, ремонт и ревизия электромонтажного оборудования холдинга;
- разработка собственных решений учета электроэнергии и мониторинга показателей деятельности холдинга «Сибзолото»;
- курирование и собственная разработка системы мониторинга промышленных приборов и транспортных средств;
- осуществление ремонтных и пусконаладочных работ на участках золотодобычи;
- сопровождение программного обеспечения систем спутникового мониторинга и мониторинга промышленных приборов на участках золотодобычи;
- информационная аналитика о деятельности транспортных средств холдинга и подрядчиков;
- предоставление отчетности о деятельности транспортных средств и промышленных приборов на участках золотодобычи.

Организационная структура в компании на 10 апреля 2019 года показана на рисунке 12.

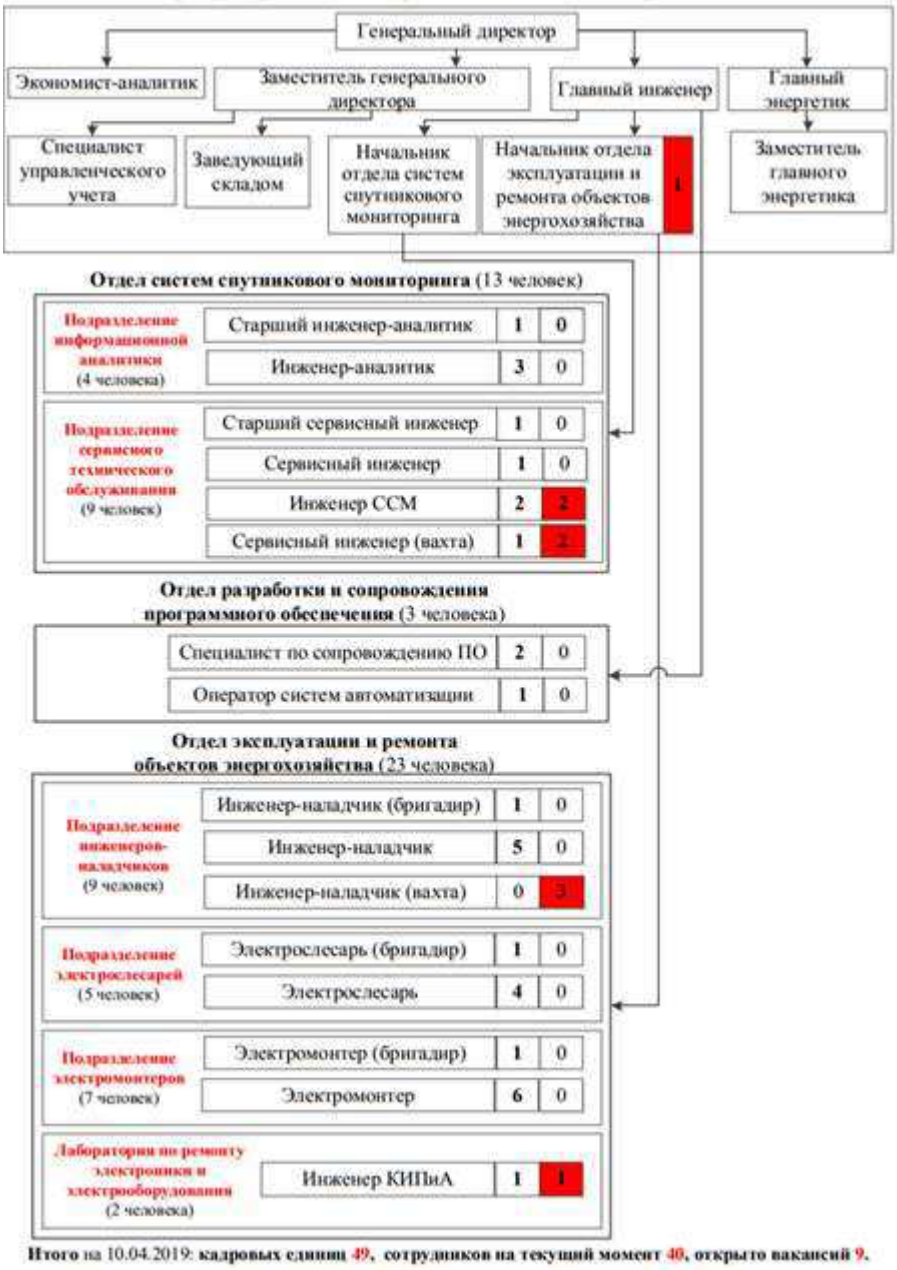


Рисунок 12 – организационная структура ООО «СпецПромАвтоматика»

Во главе предприятия находится генеральный директор, в прямом подчинении у которого находятся экономист–аналитик, заместитель генерального директора, главный инженер и главный энергетик.

Изначально ООО «СпецПромАвтоматика» сформировалась для выполнения следующих задач:

– сформировать собственный отдел эксплуатации и ремонта объектов энергохозяйства;

– взаимодействовать с компаниями, осуществляющими монтаж систем спутникового мониторинга и предоставляющие данные по парку техники;

– взаимодействовать и курировать деятельность сторонних разработчиков, разрабатывающих собственного программного обеспечения – панель мониторинга промышленных приборов;

– сформировать отдел информационной аналитики по технике холдинга.

Для мониторинга техники холдинга и подрядных организаций используется система спутникового мониторинга «АвтоГРАФ», производителем которого является компания ООО «ТехноКом».

До 2017 года монтаж модулей ССМ и их программная поддержка осуществлялась сторонними организациям – ГК «Антей» и ГК «Гугол».

С 2017 года была поставлена задача сократить затраты холдинга на монтаж и поддержку систем спутникового мониторинга. В период с 2017 года по конец 2018 года были проведены мероприятия, указанные в таблице 2.

Таблица 2 – Проведенные мероприятия для снижения себестоимости ССМ и полученные результаты

Мероприятия	Результат
Был сформирован отдел систем спутникового мониторинга, для самостоятельного монтажа и проверки оборудования, самостоятельного ремонта датчиков и бортовых контроллеров, выезды на участки с целью ревизии оборудования и проверки его работоспособности, проведение работ по тарировке баков и так далее.	Появление в организации собственного подразделения, ответственное за монтаж и ревизию оборудования ССМ.
В рамках отдела ССМ были наняты специалисты на вахтовый метод работы, для выполнения ремонтных и монтажных работ непосредственно на участках золотодобычи.	Решение позволило существенно снизить накладные расходы на командировки, а также повысить количество исправного оборудования.
Были самостоятельно установлены и настроены программные продукты компании «ТехноКом» – в корпоративной сети холдинга установлен «АвтоГРАФ Сервер», позволяющий получать данные с приборов «АвтоГРАФ», установлено аналогичное ПО на участковых серверах, которые принимают данные по Wi–Fi сети на участке золотодобычи, и ретранслирующие их на центральный сервер. Была изучена работа с ПО «АвтоГРАФ 5 PRO».	Это позволило отказаться от абонентского обслуживания сторонних компаний.
Установлена служба AutoGRAPH.NET Service, позволяющая получать данные программными средствами.	Возможность добавить в панель мониторинга, кроме мониторинга промышленных приборов, также дополнительный мониторинг техники холдинга.
Установлено web–приложение «AutoGRAPH Web», с помощью которого можно просматривать данные по объектам мониторинга без установки диспетчерского ПО на стационарный компьютер.	С появлением web–приложения появилась возможность пользоваться мобильным приложением AutoGRAPH Mobile и просматривать данные с мобильных устройств.

Окончание таблицы 2 – Проведенные мероприятия для снижения себестоимости ССМ и полученные результаты

Мероприятия	Результат
Началась работа по самостоятельному монтажу на технику подрядчиков холдинга «Сибзолото» системам спутникового мониторинга с последующим обслуживанием данного оборудования и последующим предоставлением аналитических услуг.	Позволило получить новую статью дохода предприятия.
Произведен переход на собственное обслуживание оборудования GSM сети и собственных SIM–карт, а также использование сети Wi–Fi холдинга на участках золотодобычи позволило реализовать на базе топливозаправщика мобильный сервер приёма данных.	Данное решение позволило увеличить процент стабильно передающих устройств.
Для проведения работ собрана тарировочная станция на базе транспортного средства типа Соболь.	Позволило сократить время на выполнение тарировок транспорта.

2.2 Обзор программных продуктов и информационных систем, использующихся в компании

Для осуществления своей деятельности ООО «СпецПромАвтоматика» использует различное программное обеспечение. Использование информационных технологий жизненно важно для стабильной и эффективной работы современного предприятия.

Корпоративная сеть, персональные компьютеры, центральные сервера, Wi–Fi сеть на участках золотодобычи и участковые сервера обслуживаются и поддерживают компанией ООО «КрасИнтегра», дочерней организацией холдинга «Сибзолото», которая занимается IT–инфраструктурой. Соответственно, рассматриваемое предприятие также использует услуги и оборудование компании «КрасИнтегра» для выполнения своих задач.

В компании для работы с документами, данными и почтой используется стандартный пакет Microsoft Office, в частности Microsoft Outlook, Microsoft Lync, а также Microsoft Word и Microsoft Excel.

Для общего доступа к файлам для сотрудников организации используется FTP–сервер, находящийся в локальной сети холдинга.

Кроме Lync, для коммуникации также активно используется Skype как внутри ООО «СпецПромАвтоматика», так во всем холдинге «Сибзолото».

С ноября 2018 года подразделения начали использовать облачное хранилище Google Drive для хранения справочной информации, к которой необходим общий доступ и совместные изменения в документах.

2.2.1 Панель мониторинга промышленных приборов и транспорта

Непосредственная деятельность компании в рамках золотодобывающего холдинга заключается в контроле над разработкой информационной панели диспетчеризации промышленных приборов и транспортных средств (Рисунок 13). В контексте данной деятельности ООО «СпецПромАвтоматика», в частности отдел разработки и сопровождения ПО (РСПО) осуществляет взаимодействие со сторонними разработчиками данной системы, оформляют требования и технические задания на разработку, ведут учет выполненных задач и реализованных функциональных возможностей данного продукта для последующих расчетов с разработчиками, проводят тестирование системы на предмет ошибок.

Рисунок 13 – панель мониторинга промышленных приборов и транспортных средств

Данная система разрабатывается на платформе Ensemble, разработанной компанией InterSystems. Выбор этой среды обусловлен ее способностью хранить и быстро обрабатывать большие объемы данных с помощью базы данных Cache, которая также является собственной разработкой InterSystems. Сама платформа имеет встроенную административную панель, которая удовлетворяла многие запросы предприятия, которые были заявлены при выборе средств разработки диспетчерской панели (Рисунок 14). Вся внутренняя бизнес–логика в данной системе строится на языке Cache Object Script. Пользовательская часть диспетчерской панели разрабатывается на каркасе AngularJS, как одностраничное web–приложение (Single Page Application).

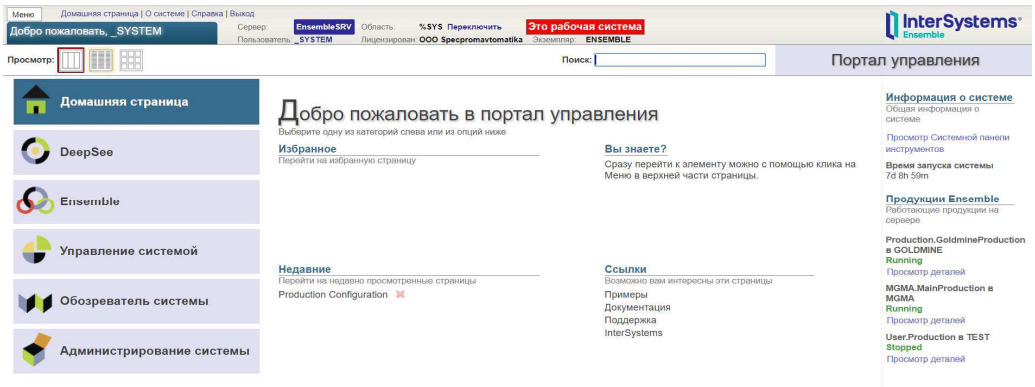


Рисунок 14 – портал управления платформой Ensemble

По состоянию на 2019 год в диспетчерской панели имеется четыре различных модуля:

– модуль «Промышленные Приборы»;
– модуль «Топливо»;
– модуль «ДЭС и ДНС»;
– модуль «Статистики ТС».

В модуле «Промышленные приборы» отображаются данные в онлайн режиме всех промышленных приборов (промывочных комплексов и

экскаваторов), которые необходимы для анализа работы участков. В данном модуле указаны следующие характеристики приборов:

1) Наличие или отсутствие связи с определенным прибором.
2) Наличие или отсутствие связи со всем участком.
3) Онлайн статус режима работы прибора (в работе, неактивен и так далее).
4) Возможность на ленточных графиках посмотреть деятельность прибора за выбранный период.
5) Линейный график используемой мощности за выбранный период.
6) Линейный график тока за выбранный период.
7) Расчет накопленной энергии прибора и перекачанной воды.
8) Просмотр количества погрузок породы на промывочный стол по времени.
9) Указание комментариев оператором панели для поддержания актуальной информации по промышленным приборам.
10) Вывод отчета по прибору за определенный период.

«СпецПромАвтоматика» кроме курирования разработкой диспетчерской панели также осуществляет монтаж и поддержку работоспособности оборудования, фиксирующего и передающего данные с этих приборов.

С физической точки зрения реализация системы передачи данных выглядит следующим образом – к прибору подключается счетчик электрической энергии «Меркурий 230», передающий данные по кабелю RS–232 на преобразователь импульсов Nport, который подключен в локальную сеть на участке.

Количество погрузок породы фиксируется с помощью нажатий на кнопку гидромониторщиком, когда производится непосредственная погрузка породы на промывочный стол. Система реализована следующим образом – к кнопке подключен счетчик импульсов СИ30 производства компании «ОВЕН», передающий данные по кабелю PS–232 в преобразователь импульсов ET–485, который подключен в локальную сеть участка.

Данные передаются по протоколу Modbus RTU, преобразователи сигналов ET–485 и Nport преобразовывают их в протокол Modbus TCP. На участковом сервере располагается служба трансмиттера, которая опрашивает приборы в сети и записывает в локальную БД на участковом сервере. Служба сендера на участковом участке забирает эти данные с локальной БД и ретранслирует их на центральный сервер, на котором расположен Ensemble и панель мониторинга.

В модуле «Топливо» указана информация о складах горюче–смазочных материалов (ГСМ) и бензовозах, работающих на участках золотодобычи. Модуль отображает последние данные по количеству топлива в баках, а также ленточные графики работы бензовоза или склада ГСМ во времени. В панель пересылаются данные о заправках транспорта данными единицами и заправки собственного бака.

В модуле «ДЭС и ДНС» отображена информация о дизельных электростанциях и дизель–насосных станциях. Кроме режима работы также указываются факты перегревов и линейные графики, отображающие работу объекта мониторинга.

В модуле «АвтоГРАФ» отображена сводная информация по всем подключенным к системам спутникового мониторинга транспортным средствам холдинга и дочерних предприятий. На главной странице данный модуль отображает следующую информацию:

1) Столбчатую диаграмму о состоянии техники по направлениям и участкам.
2) Круговую диаграмму распределения всей техники по статусам работы (исправно, неисправно, обработка, ремонт).
3) Таблицу перегревов, в которой отражены факты перегревов у техники.
4) Сводную таблицу всей техники с названием транспортного средства (ТС), закрепленным участком, гос. номером, борт. номером, датой последнего изменения статуса состояния, статус ТС, статус датчика уровня топлива (ДУТ), значение датчика температуры.

В детализации транспортного средства можно посмотреть:

1) Комментарии по состоянию ТС.
2) Ленточный график работы ТС.
3) Точечно–линейный график состояния ТС.
4) Заправки собственного бака.
5) Перегревы на ТС.
6) Количество моточасов.

Диспетчерская панель находится в статусе разработки, добавляются новые функциональные возможности.

2.2.2 Программное обеспечение систем спутникового мониторинга

Программное обеспечение для систем спутникового мониторинга используется от компании «ТехноКом», так как монтаж модулей ССМ производится от этой же компании:

- для сбора и ретрансляции данных используется АвтоГРАФ Сервер, развернутый на центральном и участковых серверах;
- диспетчерское программное обеспечение, используемое отделом информационной аналитики – ПО Автограф 5 PRO;
- для передачи данных в панель мониторинга и сторонние программные продукты используется сервис AutoGRAPH.NET;
- web–приложение «AutoGRAPH Web» для работы с данными по технике через браузер и мобильное приложение.

Так как системы спутникового мониторинга являются одним из основных направлений деятельности компании, поддержка ПО в работоспособном состоянии – приоритетная задача, поэтому необходимо более подробно описать работу данного комплекса программных средств.

В первую очередь, список объектов мониторинга, настройки фильтрации, список геозон, список водителей, атрибуты каждого объекта и т.д. настраиваются в схеме через диспетчерское ПО «АвтоГРАФ 5 PRO».

Схема – это рабочая конфигурация программы «АвтоГРАФ 5 PRO», включающая в себя: список объектов мониторинга, оснащенных контроллерами мониторинга «АвтоГРАФ» и настройки этих объектов; список геозон, водителей, инструментов (с/х и другие), настройки рабочей области программы, определенный набор модулей (панелей) программы и т.д. Каждая схема программы содержит сведения о серверах, с которых в эту схему загружаются данные.

За правильную настройку фильтрации данных объектов в схеме отвечает Главный инженер, за корректные параметры и отсутствие ошибок в схеме отдел разработки и сопровождения ПО.

AutoGRAPH Server имеет возможность создания пользователей для работы с данными по технике, а также настройки прав доступа и ограничения на просмотр данных по определенным контроллерам. Кроме этого, при административных правах доступа настроенную схему можно импортировать на сервер, чтобы у других пользователей была возможность ее скачать. Эта же схема импортируется и в службу AutoGRAPH.NET для нужного отображения данных по объектам мониторинга, и на web–клиент.

Для возможности получения данных по новым модулям «АвтоГРАФ» на собственные сервера при получении партии приборов отправляется запрос в компанию «ТехноКом» на включение их в лицензию сервера.

2.2.3 Системы организации рабочих процессов и управления задачами

В ООО «СпецПромАвтоматика» нет единой среды для управления задачами.

Отдел разработки и сопровождения ПО для осуществления своей деятельности, отслеживания и инициализации задач использует бесплатную версию Trello – программу для управления проектами небольших групп (Рисунок 15). Данное приложение использует парадигму управления проектами, известную как канбан, метод, который первоначально был популяризирован Toyota в 1980–х для управления цепочками поставок.

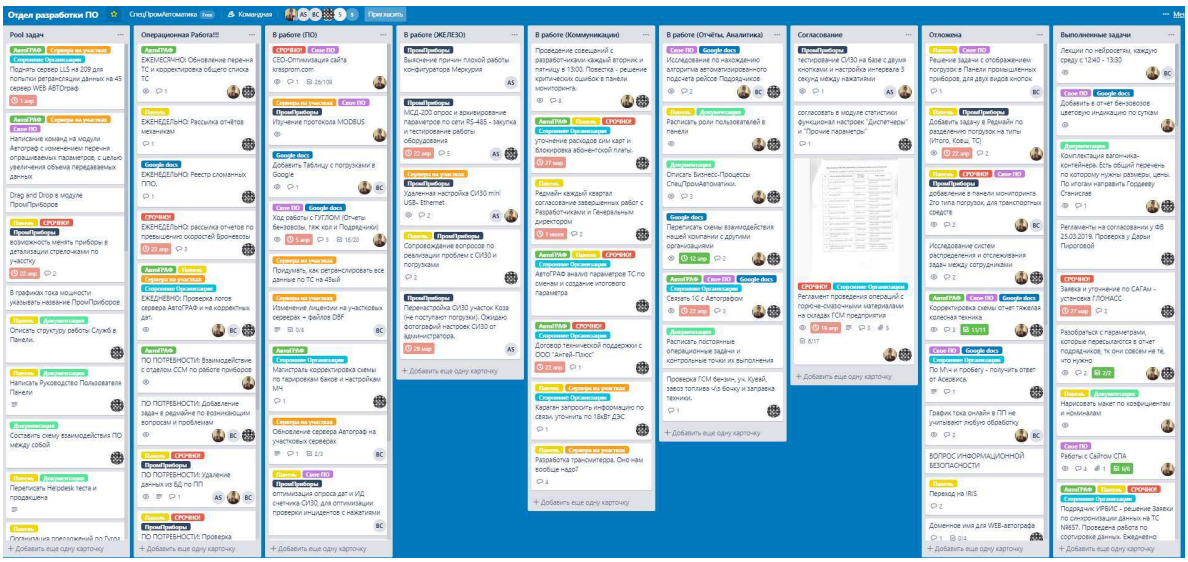


Рисунок 15 – Канбан–доска Trello

Данное решение позволяет хранить весь список задач отдела в одном месте, указывать ответственных за выполнение, вести наглядный процесс выполнения задачи, указывать сущности, связанные с этой задачей. Доска имеет несколько колонок с задачами, организация процесса отслеживания и выполнения задач выглядит следующим образом:

1) Pool задач – все новые задачи изначально попадают в данный список. Указывается название задачи, описание к задаче, а также теги, к чему относится задача.

2) Операционная работа – постоянные задачи отдела, с указанием ответственных за нее. Необходима для самоконтроля выполнения повторяющихся задач.

3) В работе (ПО) – список актуальных задач, связанные с любыми информационными продуктами.

4) В работе (ЖЕЛЕЗО) – список актуальных задач, связанные с физическими устройствами.

5) В работе (Коммуникации) – список актуальных задач, связанные с любыми коммуникациями с другими людьми, отделами, подразделениями, разработчиками и так далее.

6) В работе (Отчеты, Аналитика) – список актуальных задач, связанные с составлением любого рода отчетности, или аналитической работы, или проведением различного рода исследований.

7) Согласование – список задач, которые со стороны отдела разработки и сопровождения ПО выполнены и ожидают дальнейших согласований или подтверждений о проверки результата работы.

8) Отложена – список задач, потерявшие свою актуальность в выполнении по разным причинам (найден более оптимальный способ решения, снизился приоритет задачи и так далее).

9) Выполненные задачи – список выполненных задач. Хранится для последующей отчетности работы отдела.

Отдел ССМ использует Pyrus – онлайн–платформа управления задачами (Рисунок 16). Данное приложение было выбрано по причине большего количества необходимых функциональных возможностей в бесплатной версии для специфики отдела ССМ.

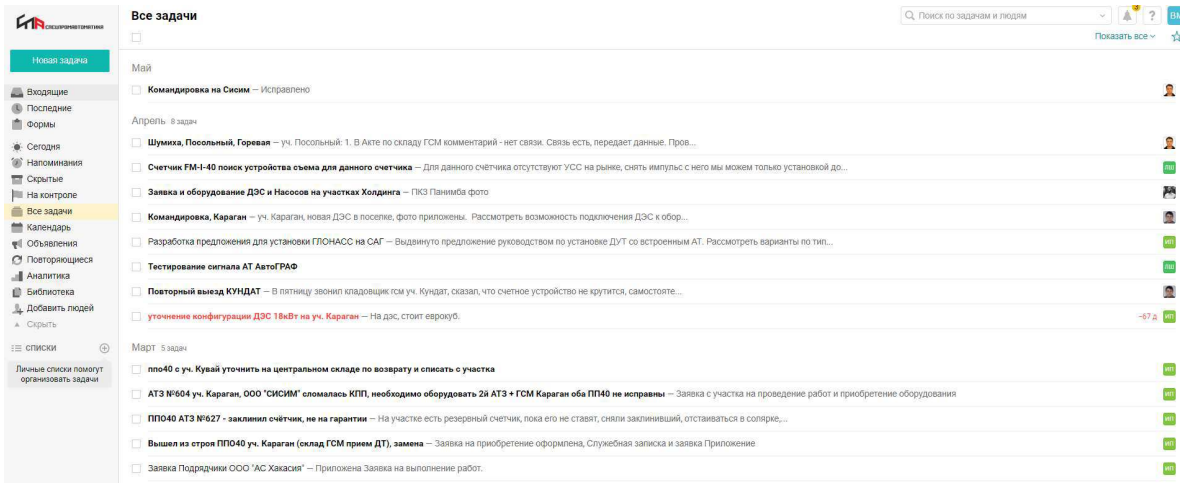


Рисунок 16 – интерфейс приложения Pyrus

Главные преимущества данного приложения для отдела ССМ:

– наличие бесплатной версии;
– наличие мобильного приложения;
– доступная и быстрая возможность постановки задачи;
– контроль выполнения задач и разбиение задачи на этапы с разными ответственными лицами;
– совместная работа с файлами.

Ограничением в бесплатной версии является возможность одновременно хранить не более 100 задач в системе. Есть возможность удалять старые задачи и тем самым не превышать лимит.

Руководящий состав использует между собой постановщик задач в почтовом клиенте Microsoft Outlook ввиду большого количества времени работы с данным почтовым клиентом и наличием регламентированного способа коммуникации в холдинге с помощью Microsoft Lync.

Для отображения прогресса разработки диспетчерской панели, курирования проекта и взаимодействия отдел РСПО совместно с разработчиками использует web–приложение для управления проектами и задачами Redmine (Рисунок 17).

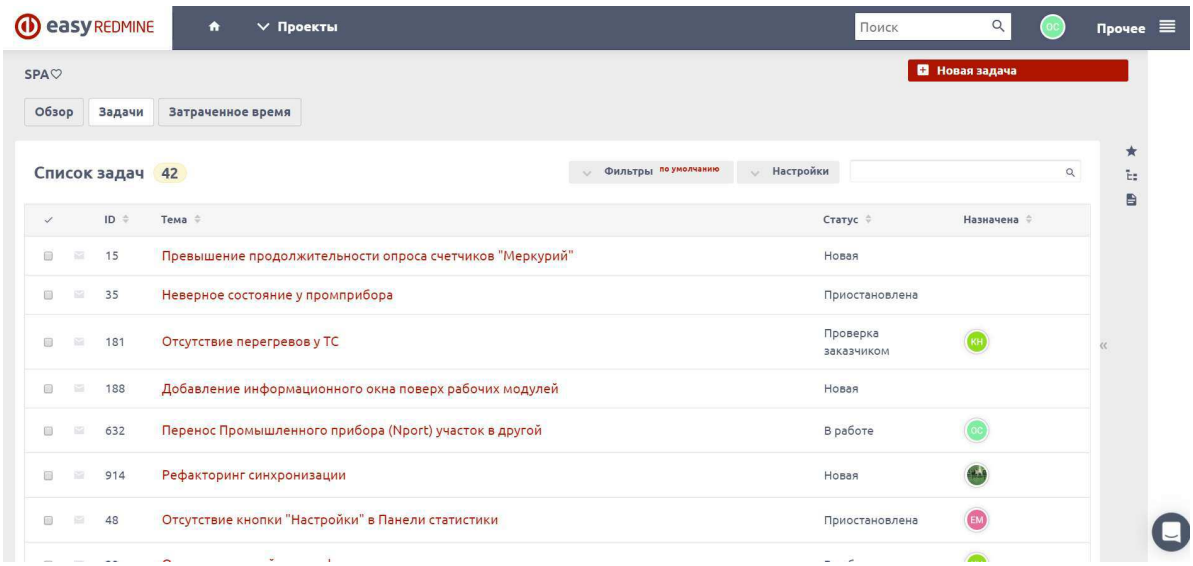


Рисунок 17 – приложение для управления проектами и задачами Redmine

Различные приложения и сервисы позволяют эффективно работать каждому отделу ООО «СпецПромАвтоматики» и своевременно выполнять свои функции.

2.3 Описание целей компании и анализ отдела информационной аналитики

Основная стратегическая цель компании – повышение экономической эффективности холдинга «Сибзолото».

Основная цель компании включает много направлений деятельности, в рамках данной работы рассматривается одна из основных целей – к концу 2019 года контролировать исправную технику холдинга, оборудованную комплектами ССМ, на уровне 400 единиц.

На рисунке 18 указано дерево целей компании.

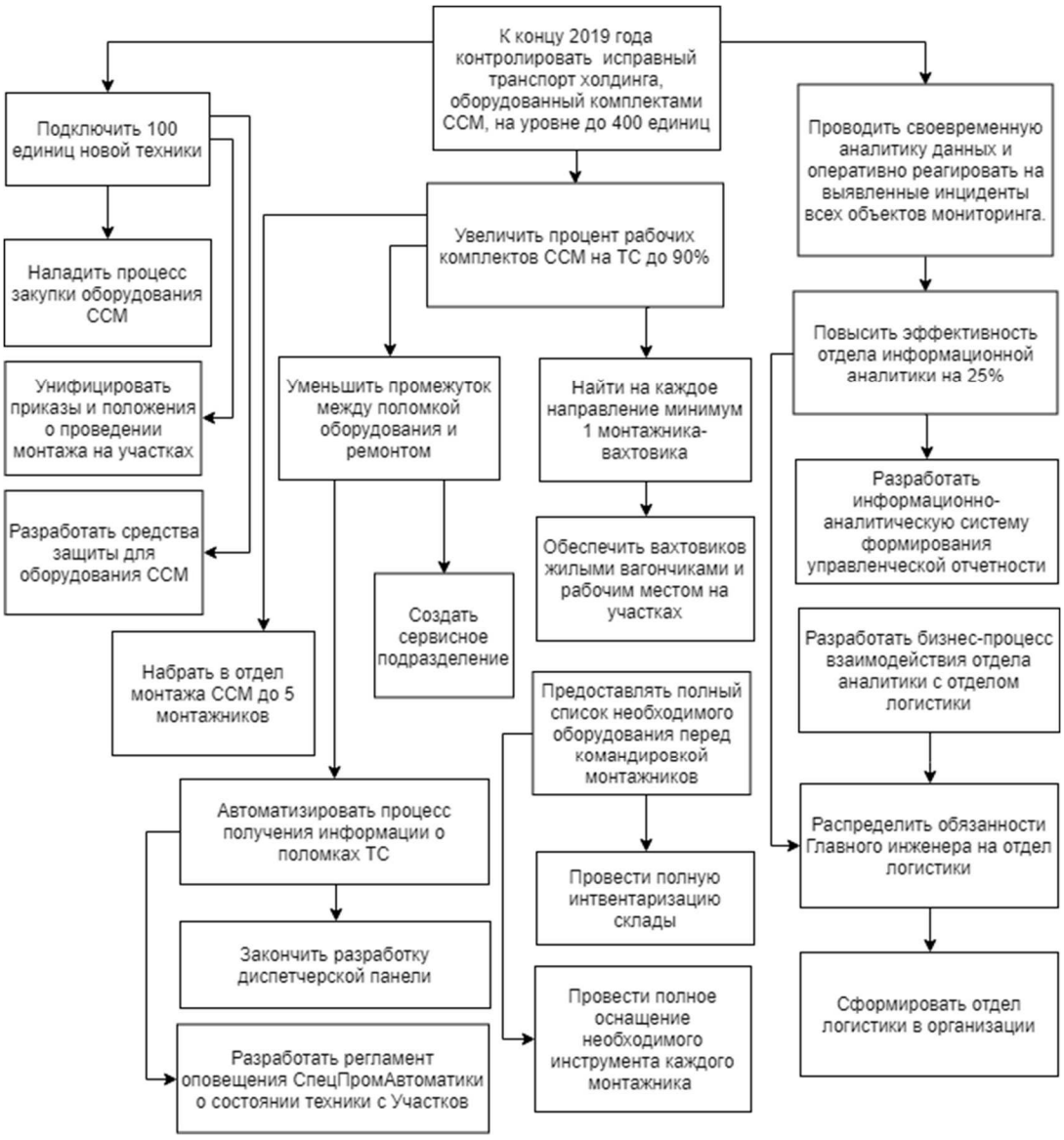


Рисунок 18 – дерево целей компании

Выделено 3 основных направления работы в рамках достижения главной задачи:

1) Подключить до конца года 100 новых единиц к системам спутникового мониторинга.

2) Увеличить количество работоспособных комплектов ССМ, уже установленные на технике до 90%.

3) Проводить своевременную аналитику техники и оперативно реагировать на выявленные несоответствия деятельности объектов мониторинга.

Кроме декомпозиции цели на задачи для ее достижения необходимо выяснить исходные данные, связанные с ней. Это поможет точно определить необходимый объём работ.

По первым двум направлениям уже были выделены задачи и подзадачи, указанные на рисунке 18. Последнее направление связано с деятельностью отдела информационной аналитики.

2.3.1 Функционально–стоимостный анализ отдела информационной аналитики

Для определения начальных условий необходимо изучить деятельность отдела информационной аналитики (ИА). Отдел состоит из четырех сотрудников – старшего аналитика и трех аналитиков. Основной функцией отдела является проведение аналитической работы по данным с модулей ССМ для следующих целей:

- выявление некорректной работы техники;
- поиск фактов недобросовестной эксплуатации техники и служебного транспорта;
- контроль использования горюче–смазочных материалов (ГСМ);
- определение факта поломки оборудования.

Результатом работы является формирование отчетности, аналитические заключения и обработка служебных записок. Деятельность отдела направлена на поиск несоответствий в работе техники и предоставлении достоверной информации о ситуации на участках золотодобычи.

Начальные условия для определения объема работ следующие:

- всего оборудовано и когда–либо передавало данные 444 единицы техники;
- передают данные 346 модуля ССМ;
- реакция на заявки от отдела информационной аналитики от проявления инцидента по началу работ по нему в среднем составляет 10 дней.

Обрабатывается сейчас 78% процентов от всей техники отделом информационной аналитики.

При изучении деятельности отдела было выявлено, что отчетность не всегда своевременно предоставляется. При существующих 346 объектах мониторинга становится очевидно, что нарастающая нагрузка отрицательно сказывается на эффективности отдела информационной аналитики.

Функционально–стоимостной анализ (ФСА) – это метод системного исследования функций объекта (изделия, процесса, структуры), направленный на минимизацию затрат при сохранении (повышении) его качества и полезности.

Структурная модель отдела информационной аналитики (ИА) показана на рисунке 19.

Рисунок 19 – структурная модель отдела информационной аналитики

В таблице 6 приведены характеристики функций отдела информационной аналитики.

Таблица 6 – Функции отдела информационной аналитики

Наименование функции			Индекс функции
Глагол	Существительное	Дополнение	
Анализировать	Данные	Холдинга	F
Составлять	Служебные записки	По выявленным несоответствиям	F1
Предоставлять	Сведения	О работоспособности оборудования	F2
Составлять	Статистику	Общую	F3
Получение	Раздаточных ведомостей		F4
Анализ	Данных	По запросам	F5

На рисунке 20 показана функциональная модель отдела.

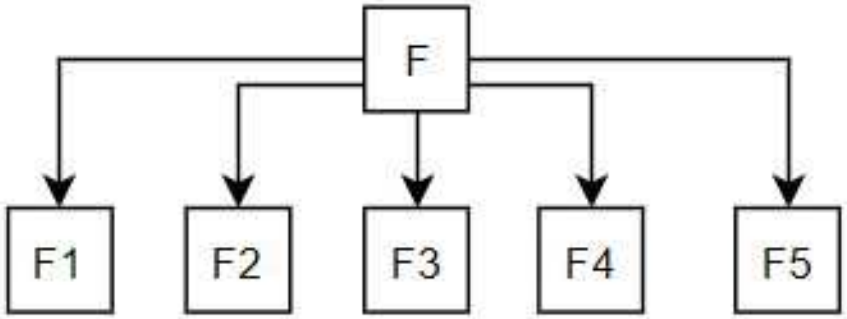


Рисунок 20 – функциональная модель отдела

Необходимо провести оценку значимости функциональной модели методом расстановки приоритетов.

Оценка значимости функций производится для следующих целей:

− построения функционально–стоимостных диаграмм и сопоставления затрат по функциям с их значимостью;

− определения допустимых затрат на функции;

− предварительной оценки качества исполнения вариантов.

Для оценки значимости функции используется экспертный метод, метод расстановки приоритетов.

Определение значимости функций этим методом осуществляется в несколько этапов:

1) Определяется количество сравниваемых функций. В рассматриваемом случае их 5.

2) Строится матрица смежности (квадратная матрица), где функции сравниваются друг с другом по предпочтению (Таблица 7).

3) Знаки < = > заменяются соответствующими коэффициентами (= на 1, > на 2, < на 0), они называются коэффициентами предпочтения. В последней графе указывается суммарное значение, подсчитанное по строке матрицы.

4) Расчет абсолютного приоритета (P_{1j}):

$$\overrightarrow{|P_j|} = |F_j| * \overrightarrow{|\sum J|}\downarrow \quad (1)$$

где $|F_j|$ – вектор строка; $\overrightarrow{|\sum J|}\downarrow$ – вектор столбец.

Относительная значимость (R_j) функций вычисляется в долях от единицы по формуле:

$$R_j = \frac{P_j}{\sum_j^m P_j} \quad (2)$$

где P_j – абсолютный приоритет j–ой функции; m – количество функций.

Таблица 7 – Матрица смежности функций отдела

	F1	F2	F3	F4	F5
F1	=	>	>	<	<
F2	<	=	>	<	<
F3	<	<	=	<	<
F4	>	>	>	=	<
F5	>	>	>	>	=

В качестве экспертов выступали старший аналитик, главный инженер и экономист–аналитик. Оценка главных функций представлена в таблице 8.

Таблица 8 – Оценка главных функций

	F1	F2	F3	F4	F5	Сумма	Абсолютная значимость	Относительная значимость
F1	1	2	2	0	0	5	13	15,29%
F2	0	1	2	0	0	3	5	5,88%
F3	0	0	1	0	0	1	1	1,18%
F4	2	2	2	1	0	7	25	29,41%
F5	2	2	2	2	1	9	41	48,24%
Итого:							85	100,00%

Перед построением совмещенной модели необходимо найти временные затраты отдела на каждую из функций. В таблице 9 указаны: формируемая отчетность отдела информационной аналитики, затраты времени каждого аналитика на формирование отчета, индекс функции, к которой данный отчет относится и количество часов в месяц на его формирование. Информация в данной таблице получена из годового отчета ООО «СпецПромАвтоматика» за 2018 год.

Таблица 9 – Список всех отчетов отдела информационной аналитики

№	Название отчёта	Количество часов, потраченное на формирования данного отчета определенного сотрудника, час				Индекс функции отдела	Итого времени всего отдела на отчет, час
		Старший аналитик	Аналитик 1	Аналитик 2	Аналитик 3		
1	Накопительный отчёт по использованию карт АТЗ	6	0	0	0	F2	6
2	Сводный отчёт по использованию карт АТЗ	4	0	0	0	F3	4
3	Накопительный отчёт о работе служебного транспорта	0	6	0	0	F1	6
4	Накопительный отчёт о работе датчиков температуры	0	0	0	13	F2	13
5	Сводный отчёт о работе служебного транспорта	8	0	0	0	F3	8
6	Отчёт о заправках техники АТЗ	25	0	45	0	F1	70
7	Сводный отчёт о работе АТЗ и складов ГСМ	12	0	0	0	F2	12
8	Сводный отчёт о работе датчиков температуры	5	0	0	18	F2	23
9	Сводный отчёт о работе СКДШ	5	0	0	14	F2	19
10	Отчёт по технике Подрядчиков	0	108	0	0	F1	108
11	Аналитическая записка по технике Подрядчиков	0	12	0	0	F3	12
12	Отчёт по Колёсной и Тяжёлой технике Холдинг «Сибзолото»	0	0	96	0	F1	96
13	Аналитическая записка по Колёсной и Тяжёлой технике Холдинг «Сибзолото»	2	0	4	0	F2	6
14	Сводный отчёт по ДЭС и ДНС	8	0	0	24	F1	32
15	Отчёт по пробегу служебных транспортных средств	16	15	0	0	F1	31
16	Фотоотчёт по СКДШ накопительный	0	0	0	13	F2	13
17	Отчёт по скорости служебного транспорта	0	5	0	0	F1	5
18	Отчёт по скорости Тяжёлой и колёсной технике	0	0	4	0	F1	4
19	Отчёт по работе мобильных модулей	10	0	0	0	F2	10
20	Управляющая таблица по работе СКДШ	8	0	0	0	F3	8
21	Отчет по складам ГСМ	5	0	0	38	F1	43

В таблице 10 указано суммарное время каждого сотрудника на выполнение функций №1, №2 и №3 и времени всего отдела на каждую функцию.

Таблица 10 – Время сотрудников и всего отдела на функции

Индекс функции	Рабочее время сотрудника, часы				Использовано времени всего отдела, часы
	Старший аналитик	Аналитик 1	Аналитик 2	Аналитик 3	
F1	54	134	145	62	395
F2	40	0	4	58	102
F3	20	12	0	0	32
Итого времени сотрудника:	114	146	149	120	**529**

Функцию F4 (получение раздаточных ведомостей) выполняет аналитик 3 и это занимает у него 10 часов рабочего времени в месяц.

Для функции F5 (анализ данных по запросам) остается все остальное свободное рабочее время отдела.

На рисунке 21 показан график, иллюстрирующий, как много тратит сотрудник своего рабочего времени на выполнение определенной функции отдела.

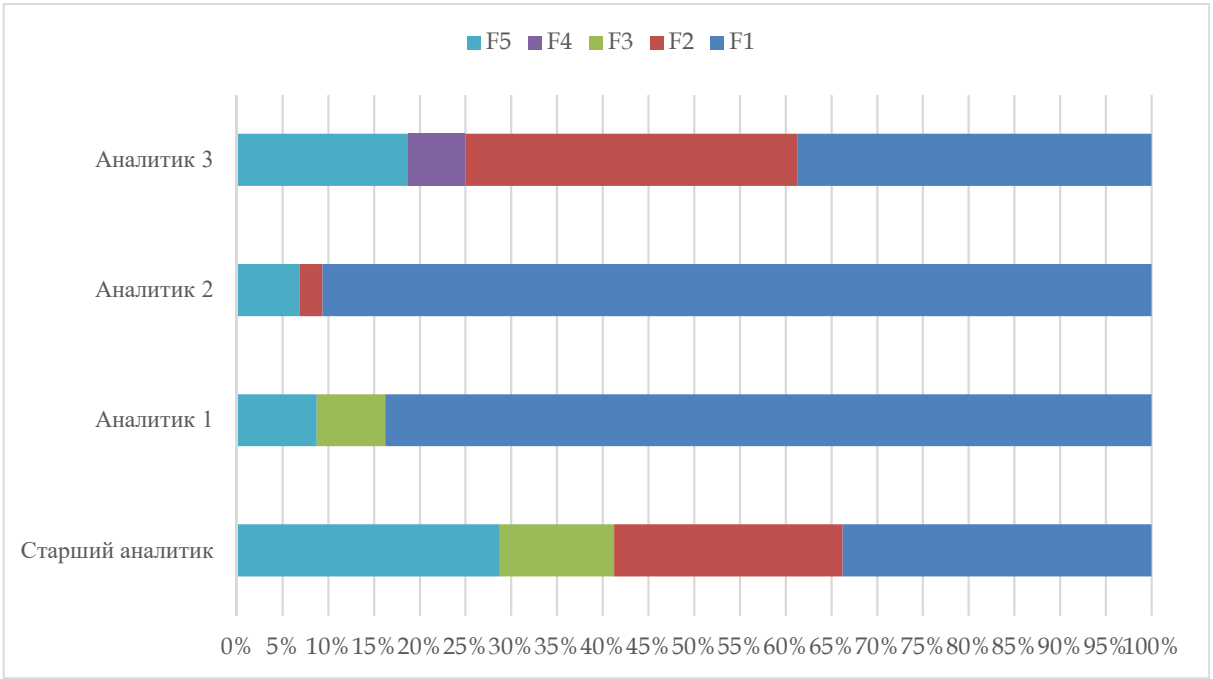


Рисунок 21 – линейчатый график распределения времени сотрудников на функции отдела

В качестве исходной информации при построении совмещенной модели используются структурная модель и функциональная модель. Построение происходит путем наложения функциональной модели на структурную. Совмещенная модель представлена в таблице 11.

Таблица 11 – Совмещенная модель

Сотрудник	Оклад, тыс. руб.	F1	F2	F3	F4	F5
Старший аналитик	55	18,5625	13,75	6,875	0	15,8125
Аналитик 1	40	33,5	0	3	0	3,5
Аналитик 2	40	36,25	1	0	0	2,75
Аналитик 3	40	15,5	14,5	0	2,5	7,5
Затраты на осуществление функции	175	103,8125	29,25	9,875	2,5	29,5625
Относительные затраты на осуществление функции		59,32%	16,71%	5,64%	1,43%	16,89%

Строим функционально–стоимостную диаграмму. Сопоставление оценок с помощью диаграммы «значимость (важность) – затраты» определяется по формуле:

$$K_j = \frac{Z_j}{R_j} \quad (3)$$

где, Z_j – относительные затраты на осуществление j–ой функции;

R_j – относительная важность j–ой функции.

Функциями, с неблагоприятным соотношением «затраты – важность (значимость)» будут считаться те, у которых $K_j > 1$.

В таблице 12 представлены значения коэффициента «затраты – важность».

Таблица 12 – Значения коэффициента «затраты–важность»

Индекс функции	Относительная значимость функции (в долях)	Относительные затраты функции (в долях)	Коэффициент «значимость (важность) – Затраты»
F1	0,1529	0,5932	3,88
F2	0,0588	0,1671	2,84
F3	0,0118	0,0564	4,80
F4	0,2941	0,0143	0,05
F5	0,4824	0,1689	0,35

На рисунке 22 графическое представление таблицы 12.

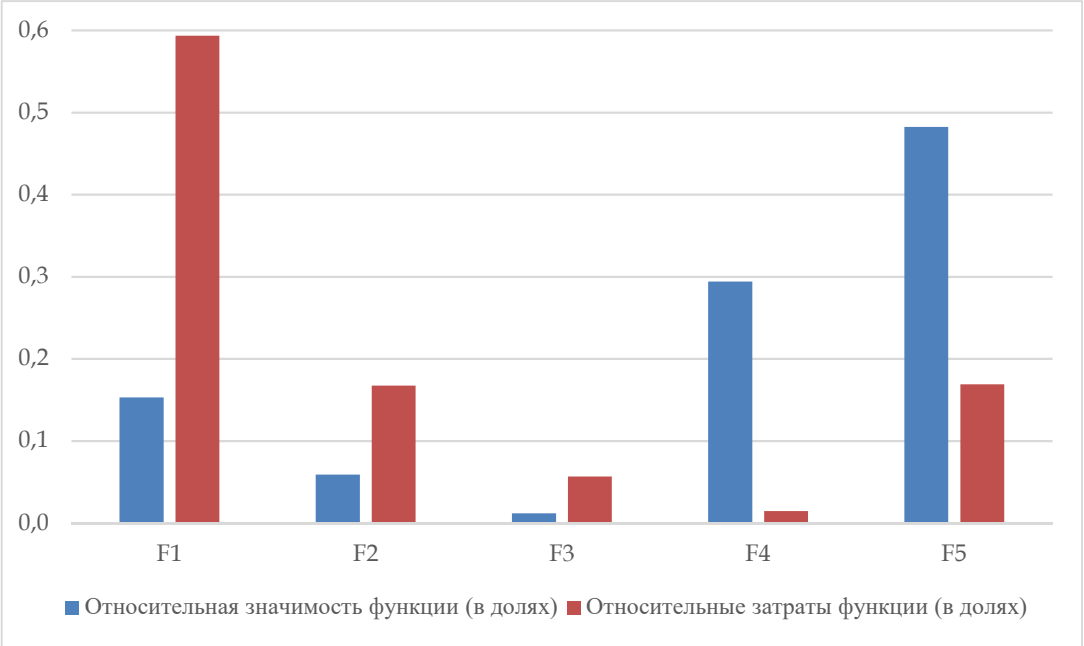


Рисунок 22 – гистограмма важности и значимости функций

В результате применения функционально–стоимостного анализа подтверждено, что необходимо детально рассмотреть процесс формирования отчетности в отделе. Из–за большого количества отчетов выберем для анализа отчеты определенной функции. Хотя наибольший коэффициент «значимость–важность» наблюдается у функции F3, правильным решением будет рассмотреть функцию F1, так как именно она наиболее затратна для выполнения, а в рамках задачи компании, указанной в пункте 2.3, необходимо в

первую очередь снизить нагрузку на отдел, а не уровнять затраты на функции отдела. Для дальнейшего анализа функции F1 используем АВС–анализ и диаграмму Парето.

2.3.2 АВС–анализ формирования отчетности

Для дальнейшего анализа затрат из таблицы 8 оставим только отчеты, формирование которых выполняет F1, а также преобразуем рабочее время сотрудников на формирование данного отчета в денежный эквивалент себестоимости отчета для ООО «СпецПромАвтоматика». Для этого воспользуемся формулой:

$$\text{Стоимость отчета} = \frac{\text{Заработная плата в руб.}}{\text{кол} - \text{во рабочих часов в мес.}} * \text{Потраченное время} \quad (4)$$

Для проведения АВС–анализа необходимо построить диаграмму Парето с помощью данных из таблицы 13. Информация из таблицы является базовой для построения диаграммы Парето и АВС – анализа.

Построение диаграммы начинают с наиболее затратных элементов. Каждая точка кривой представляет затраты по всем предшествующим элементам нарастающим итогом.

На графике выделяют три зоны:

- A, соответствует 75 % всех затрат;
- B, соответствует 20 % всех затрат;
- C, оставшимся 5 %.

Таким образом, в группу «А» попадают структурные элементы с наибольшими затратами. Они являются наиболее перспективными с точки зрения проведения анализа.

В зону «В» попадают элементы, имеющие некоторый резерв для снижения затрат. В зоне «С» резервов не существует. Диаграмма Парето по стоимости отчетов и зонами для АВС–анализа на рисунке 23.

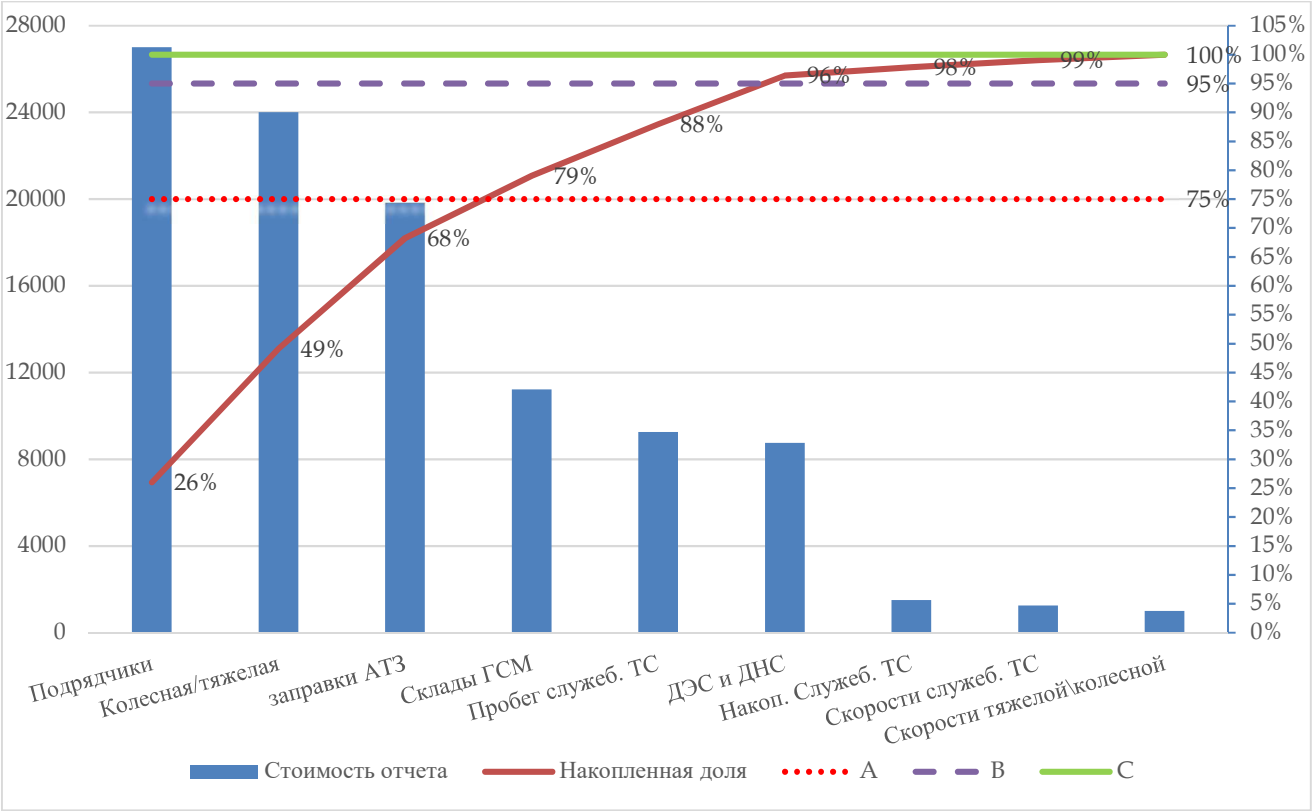


Рисунок 23 – Диаграмма Парето с ABC–зонами

В A–зону с наибольшими затратами попали следующие отчеты:

- отчёт по технике Подрядчиков;
- отчёт по Колёсной и Тяжёлой технике Холдинга «Сибзолото»;
- отчёт о заправках техники АТЗ.

На формирование этих отчетов суммарно используется 70843,75 рублей в месяц, что составляет 68% всех расходов на функцию составления служебных записок по найденным несоответствиям. Необходимо описать существующие бизнес–процессы формирования данных отчетов для определения проблемных зон.

2.4 Описание существующих бизнес–процессов формирования отчетности

Для описания бизнес–процессов выбран язык нотации BPMN – система условных обозначений (нотация) и их описания в XML для моделирования бизнес–процессов. Разработана компанией Business Process Management Initiative и поддерживается компанией Object Management Group, после слияния обеих организаций в 2005 году.

Моделирование в BPMN осуществляется посредством диаграмм с небольшим числом графических элементов. Это помогает пользователям быстро понимать логику процесса. Основные категории элементов:

1) Объекты потока управления:

   a. события – изображаются окружностью и означают какое–либо происшествие в мире. События инициируют действия или являются их

результатами. Согласно расположению в процессе события могут быть классифицированы на начальные, промежуточные и завершающие;

b. действия – изображаются прямоугольниками со скругленными углами. Среди действий различают задания и подпроцессы. Графическое изображение свёрнутого подпроцесса снабжено знаком плюс у нижней границы прямоугольника;

c. логические операторы (развилки) – изображаются ромбами и представляют точки принятия решений в процессе. С помощью логических операторов организуется ветвление и синхронизация потоков управления в модели процесса.

2) Соединяющие объекты:

a. поток управления – изображается сплошной линией, оканчивающейся закрашенной стрелкой. Поток управления задаёт порядок выполнения действий;

b. поток сообщений – изображается штриховой линией, оканчивающейся открытой стрелкой. Поток сообщений показывает, какими сообщениями обмениваются участники;

c. ассоциации – изображаются пунктирной линией, заканчивающейся стрелкой. Ассоциации используются для ассоциирования артефактов (данных или текстовых аннотаций) с объектами потока управления.

3) Роли:

a. пулы – изображаются прямоугольником, который содержит несколько объектов потока управления, соединяющих объектов и артефактов;

b. Дорожки – представляют собой часть пула. Дорожки позволяют организовать объекты потока управления, связывающие объекты и артефакты.

4) Артефакты:

a. Данные – показывают читателю, какие данные необходимы действиям для выполнения и какие данные действия производят.

b. Группы – изображается прямоугольником с закругленными углами, граница которого — штрихпунктирная линия. Группа позволяет объединять различные действия, но не влияет на поток управления в диаграмме.

c. текстовые аннотации – используются для уточнения значения элементов диаграммы и повышения её информативности.

Элементы этих четырёх категорий позволяют строить простейшие диаграммы бизнес–процессов.

Моделирование бизнес–процессов используется для донесения широкого спектра информации до различных категорий пользователей. Диаграммы бизнес–процессов позволяют описывать сквозные бизнес–процессы, но в то же время помогают читателям быстро понимать процесс и легко ориентироваться в его логике.

Для снижения визуальной нагрузки при описании бизнес–процессов, правило нотации, которое говорит, что развилки должны быть использованы либо для разделения потока, либо для его слияния, и не могут выполнять оба

действия, будет игнорироваться. В описанных далее бизнес–процессах логические элементы могут одновременно как быть логическим оператором разделения потока, так и соединять разные потоки в один. Данное допущение используется только при условии, что и синхронизации потоков, и разделение их производится по одному логическому правилу, указанному в операторе. На рисунке 24 показан пример с логическим оператором «или\или». В случае, если для разделения потоков и синхронизации необходима разная логика, необходимо использовать разные логические операторы (Рисунок 25).

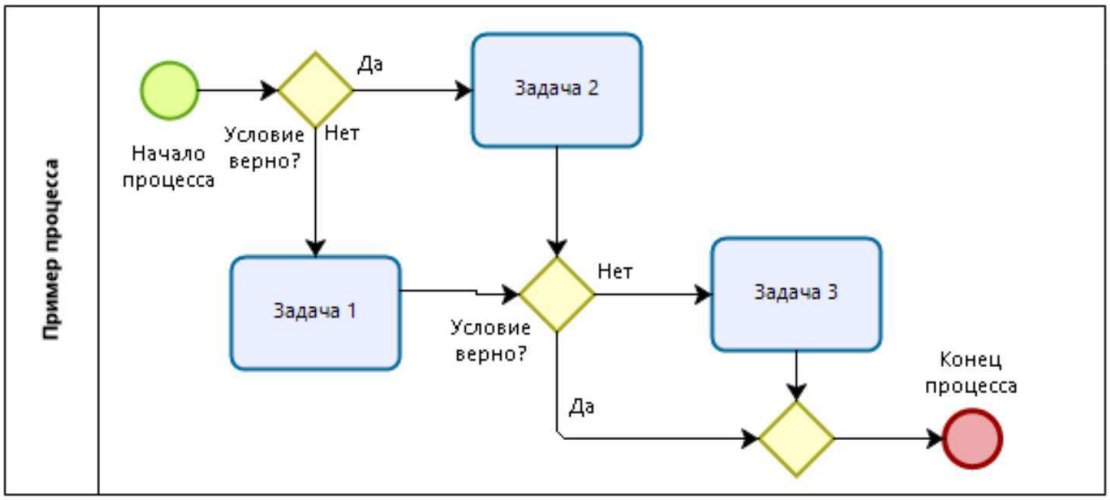


Рисунок 24 – Пример процесса с опущением правила использования двух операторов в случае одновременного слияния и распараллеливания процессов

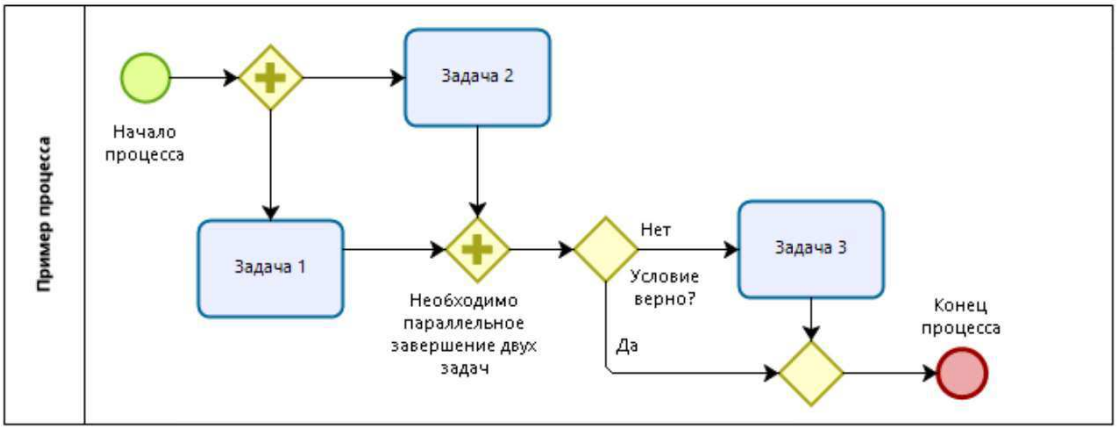


Рисунок 25 – Пример процесса, где правило использования двух операторов игнорировать нельзя

Информация о существующих бизнес–процессах формирования отчетности была получена с помощью визуального наблюдения за работой отдела информационной аналитики и опросе сотрудников.

Для подсчета времени проведения типовых операций, участвующих в бизнес–процессах, измерения проводились несколько раз и выбиралось среднее время выполнения операции.

2.4.1 Описание бизнес–процесса формирования отчета по подрядчикам

В сформированном отчете по подрядчикам располагается информация о работе всех оборудованных ССМ единицах техники подрядчиков холдинга «Сибзолото» за календарный месяц. Отчет представлен в виде книги Excel, в каждом листе которого содержатся отдельные таблицы со всей техникой по одному подрядчику (Рисунок 26).

По результатам работы с этим отчетом создается аналитическая записка в Excel, содержащая актуальную сводную информацию о деятельности подрядчиков.

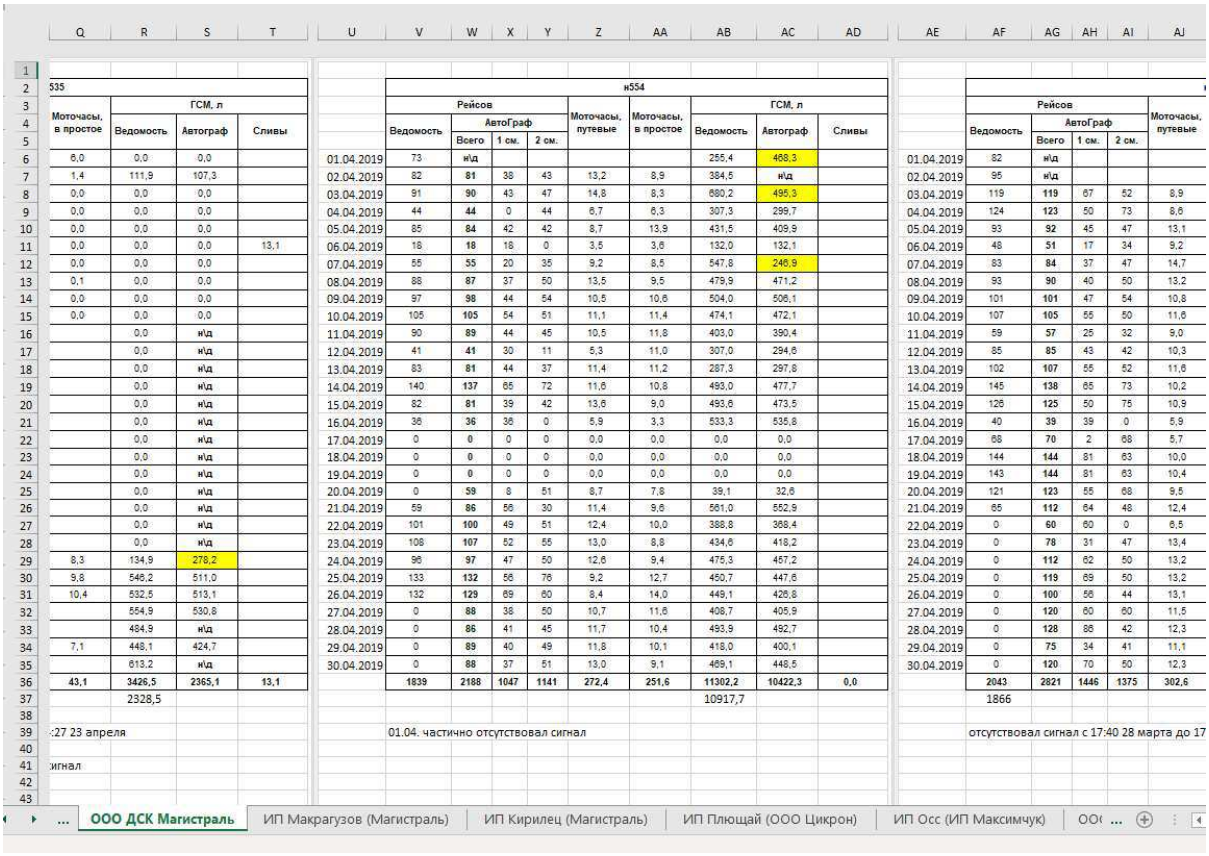


Рисунок 26 – Внешний вид отчета по подрядчикам

В данном отчете указывается следующие данные:

- количество совершенных рейсов из ведомостей подрядчиков;
- количество совершенных рейсов первой и второй смены по данным из «АвтоГРАФ»;
- путевые моточасы за две смены из «АвтоГРАФ»;
- моточасы в простое техники за две смены из «АвтоГРАФ»;
- объем заправленного топлива за сутки из ведомостей подрядчиков;
- объем заправленного топлива за сутки из «АвтоГРАФ»;
- объем слитого топлива за сутки из ПО АвтоГРАФ.

На конец мая 2019 года в отчете фигурировало 70 единиц техники и 16 подрядчиков. Из всего парка техники некорректно работает 5 датчиков уровня топлива (ДУТ).

Предоставление отчета за прошлый отчетный период происходит обычно после 15 числа следующего месяца. Завершение формирования отчета может происходить вплоть до даты сдачи отчета. Такая продолжительность связанна с ожиданием получения данных с модулей ССМ. Работа с новым отчетом начинается обычно в начале первой недели месяца.

В начале каждого месяца происходит создание и форматирование отчета за новый отчетный период. Для этого специалист сверяется с аналитической запиской по технике подрядчиков.

Этапы создания отчета указаны в таблице 14.

Таблица 14 – Список операции для формирования отчета по подрядчикам

Задания	Время задания, сек.	Частота повторения задания	Формула расчета повторов	Кол–во	Всего времени сек.
Создать новый файл отчета	2400	Один раз в месяц	Один раз	1	2400
Выбрать подрядчика и проверить наличие данных	5	В среднем 17 дней работы по отчету	17 дн.* 16 подрядчиков	272	1360
Выбрать диапазон данных по сменам	12	При наличии данных в период смен	30 дн. * 70 ТС	2100	25200
Подсчитать рейсы за смены	120			2100	252000
Перенести моточасы в отчет	8			2100	16800
Выбрать диапазон данных за сутки	10	При наличии данных за сутки		2100	21000
Провести анализ графика топлива	30	При наличии данных за сутки и проблемы с ДУТом	30 дн. * 5 ДУТ	150	4500
Найти ведомость техники	25	При работе с ведомостями по каждому ТС	3 новые ведомости * 70 ТС	210	5250
Перенести данные из ведомости в отчет за сутки	20	При наличии данных из ведомостей	30 дн. * 70 ТС	2100	42000
Итого на создание отчетов					370510

По данным из таблицы 14 аналитик формирует отчет за 102 часа, 55 минут и 10 секунд (103 часа). Полученное значение меньше указанного в таблице 9 на 5 часов.

Бизнес–процесс формирования отчета по подрядчикам холдинга «Сибзолото» в нотации BPMN описан на рисунке 27, подпроцесс заполнения отчета данными из «АвтоГРАФ» на рисунке 28.

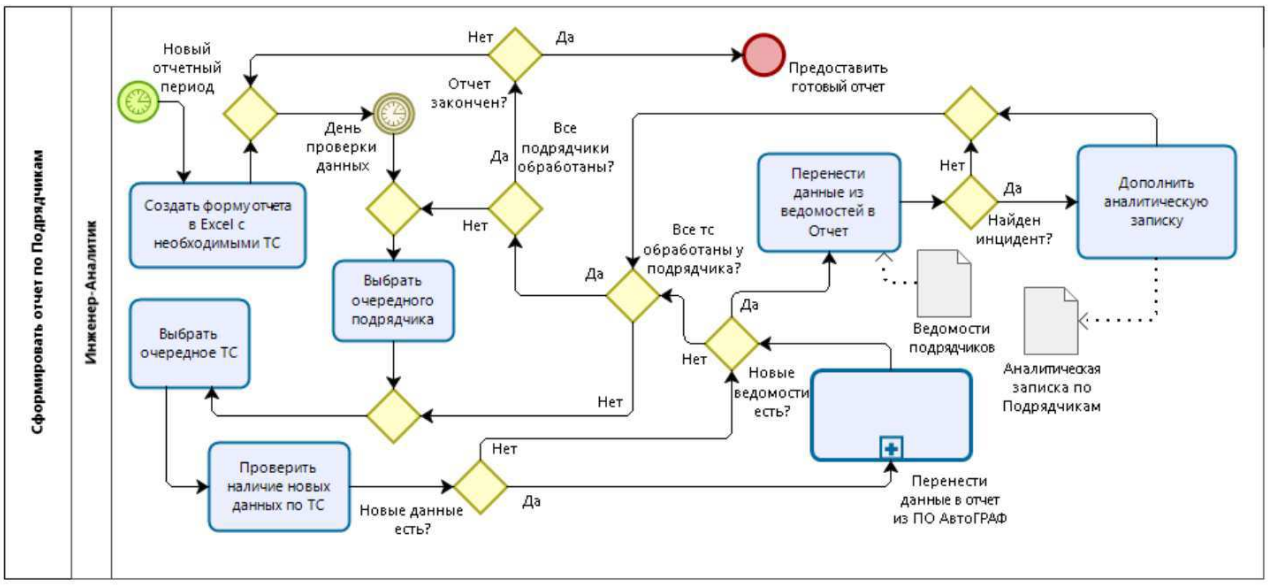


Рисунок 27 – Бизнес–процесс формирования отчета по подрядчикам

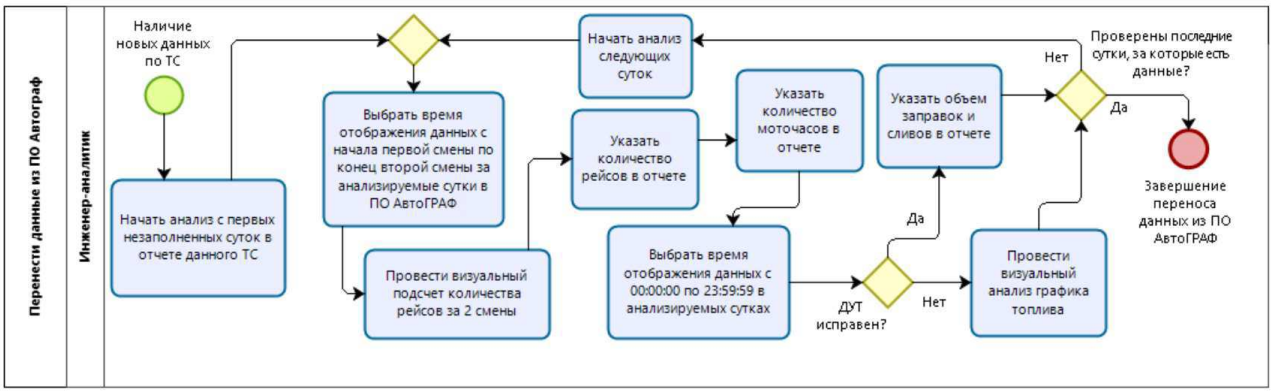


Рисунок 28 – подпроцесс переноса данных из «АвтоГРАФ» в отчет подрядчиков

Начало основного бизнес–процесса происходит в первый день нового месяца, в котором сразу создается новый пустой отчет. Промежуточное событие – начало рабочего времени аналитика, выделенного на работу с этим отчетом. Завершением процесса является предоставление готового отчета.

Подпроцесс переноса данных из «АвтоГРАФ» начинается при наличии новых данных по технике, и завершается после того, как все новые данные перенесены в отчет.

В описании бизнес–процесса также указано, что для его функционирования необходимо работать с двумя другими документами – ведомостями подрядчиков и аналитической запиской по подрядчикам.

В таблице не указаны операции работы с аналитической запиской при наличии их в бизнес–процессе. В таблице 9 аналитическая записка присутствует как отдельный отчет и на нее выделено рабочее время.

2.4.2 Описание бизнес–процесса формирования отчета по колесной и тяжелой технике холдинга «Сибзолото»

В этом отчете отображается накопительная информация о деятельности колесной и тяжелой технике холдинга «Сибзолото» с итоговыми данными за календарную неделю. Отчет представлен в виде нескольких файлов Excel на ftp–сервере (объекты мониторинга разделены на тип и направление золотодобычи), в которые добавляются данные по технике накопительным способом. После обработки данных за отчетную неделю по всем направлениям и типам техники составляется аналитическая записка по выявленным инцидентам. В данном отчете указываются следующие данные:

- дата и номер смены;
- объем заправки за смену из ПО «АвтоГРАФ»;
- объем заправок по ведомости;
- разница в объемах заправок;
- расход топлива за сутки из ПО «АвтоГРАФ»;
- сливы топлива из ПО «АвтоГРАФ»;
- моточасы из ПО «АвтоГРАФ» за неделю;
- моточасы из ведомостей за неделю.

Пример заполненного отчета показан на рисунке 29.

Модель:LW-500 Номер ТС:383 (Тюхтерек)						
Дата		ДТ, ведомость	ДТ, АвтоГРАФ	ДТ, разница	Расход ДТ	Сливы ДТ
07.06.2018	1 смена	70	82	12	250,3	
	2 смена	107	106,3	-0,7		
08.06.2018	1 смена	129	113,5	-15,5	238,1	
	2 смена	118	117,2	-0,8		
09.06.2018	1 смена	0	0	0	228	
	2 смена	221	217,5	-3,5		
10.06.2018	1 смена	161	152,3	-8,7	175	
	2 смена	0	0	0		
11.06.2018	1 смена	187	185,9	-1,1	225,6	
	2 смена	0	0	0		
12.06.2018	1 смена	216	202,3	-13,7	119,6	
	2 смена	0	0	0		
13.06.2018	1 смена	139	141,2	2,2	316,1	
	2 смена	178	176,8	-1,2		
14.06.2018	1 смена	123	120,2	-2,8	254,7	
	2 смена	124	122,9	-1,1		
15.06.2018	1 смена	155	156,1	1,1	92,8	
	2 смена	0	0	0		
16.06.2018	1 смена	0	0	0	94,5	
	2 смена	52	65,1	13,1		
17.06.2018	1 смена	131	130,6	-0,4	273,8	
	2 смена	140	131,9	-8,1		
18.06.2018	1 смена	119	120,2	1,2	69,7	
	2 смена	0	0	0		
19.06.2018	1 смена	0	0	0	18,6	
	2 смена	0	0	0		

Период контроля моточасо	9.06.2018 утро	13.06.2018 вечер	итого
Моточасы по АвтоГРАФ:			106,2
Моточасы по ведомости:	27022	27123	101
Разница:			5,2

Период контроля моточасо	14.06.2018 утро	20.06.2018 вечер	итого
Моточасы по АвтоГРАФ:			64,7
Моточасы по ведомости:	27135	27198	63
Разница:			1,7

Рисунок 29 – Внешний вид таблицы объекта мониторинга в отчете колесной и тяжелой техники

Список необходимых заданий для обработки всех отчетов по направлениям и типам техники указан в таблице 15. Так как отчет имеет накопительный характер, рассматривалось время на формирование отчетности за месяц или за 4 отчетных периода по отчету. В данных отчетах обрабатывается в среднем 140 единиц техники.

Таблица 15 – Список операции, для формирования отчета по колесной и тяжелой технике

Задания	Время выполнения задания в секундах	Частота повторения задания	Расчет количества повторов задания	Количество повторов	Время с повторами задания в сек.
Проверить наличие новых данных	4	Каждый день по всем ТС	140 ТС * 30 дней	4200	16800
Запросить данные за смену	8	Дважды за день по всем ТС	140 ТС * 2 раза за сутки * 30 дней	8400	67200
Перенести данные заправок и сливов за смену	12			8400	100800
Перенести данные расхода ДТ	6	Каждый день по всем ТС	140 ТС * 30 дней	4200	25200
Запросить данные за неделю	15	4 раза в месяц по всем ТС	140 ТС * 4 недели	560	8400
Перенести моточасы	8	4 раза в месяц по всем ТС	140 ТС * 4 недели	560	4480
Найти ведомость по ТС	35	Три раза новые ведомости, по всем тс.	140 * 3 получения ведомостей	560	19600
Заполнить отчет данными из ведомости за 1 неделю	60	4 раза в месяц на все ТС	140 ТС* 4 недели	560	33600
Продлить лист техники	30	2 раза в месяц на все ТС	140 ТС * 2	280	8400
Посмотреть график работы техники за отчетную неделю	65	4 раза в месяц на все ТС	140 ТС * 4 недели	560	36400
Итого					320880

По данным из таблицы 15 аналитик формирует отчет за 89 часов и 8 минут (89 часов). Полученное значение меньше указанного в таблице 9 на 7 часов.

Бизнес–процесс формирования отчета по колесной и тяжелой технике холдинга «Сибзолота» описан на рисунке 30, подпроцесс взаимодействия с ПО «АвтоГРАФ» на рисунке 31.

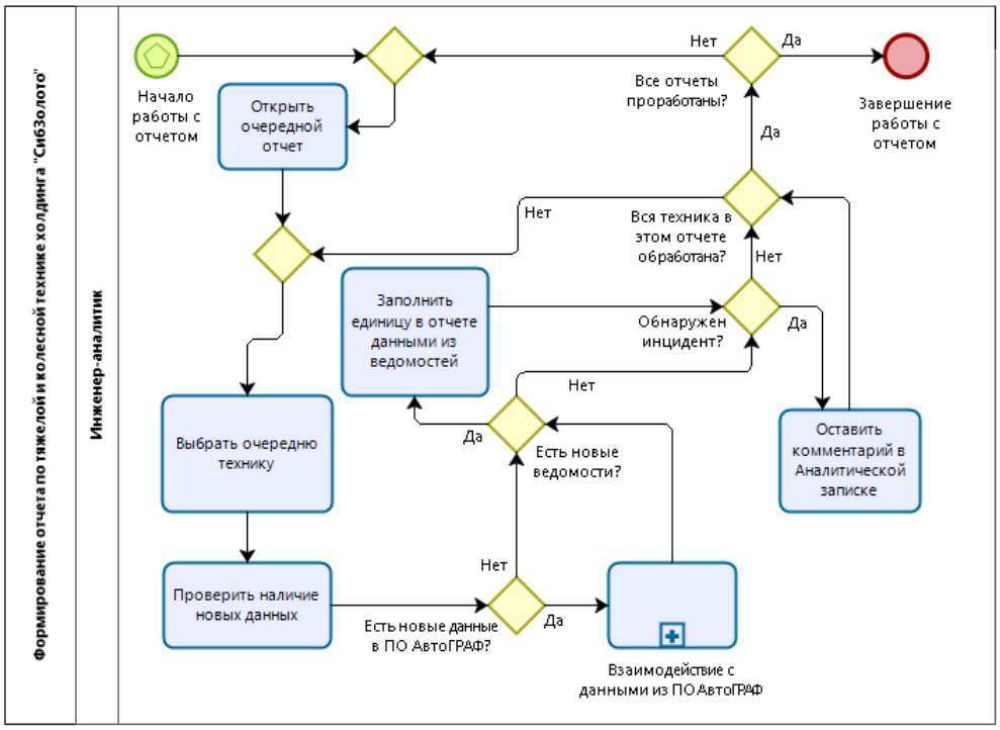


Рисунок 30 – Бизнес–процесс формирования отчетности по колесной и тяжелой технике холдинга «Сибзолото»

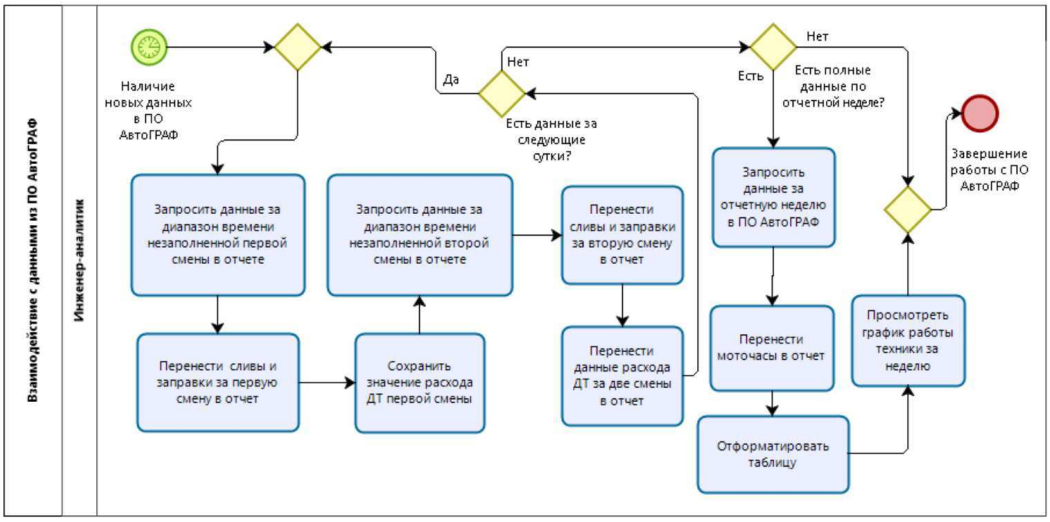


Рисунок 31 – Подпроцесс взаимодействия с ПО «АвтоГРАФ» при формировании отчета по тяжелой и колесной технике

Начало основного бизнес–процесса наступает при разных обстоятельствах – получения новых ведомостей, начало новой отчетной недели, выделено время аналитиком для работы с этим отчетом и так далее. Завершением процесса является окончательная обработка новых данных по всей технике в отчете.

Подпроцесс взаимодействия с ПО «АвтоГРАФ» начинается при наличии новых данных по технике, и завершается после того, как все новые данные перенесены в отчет.

В таблице не указаны операции работы с аналитической запиской при наличии их в бизнес–процессе по той же причине, что и при описании бизнес–процесса формирования отчета по подрядчикам.

2.4.3 Описание бизнес–процесса формирования отчета о заправках техники АТЗ

Отчет содержит подробную информацию о заправках техники АТЗ. Формируются 2–3 раза в неделю, по ним фиксируется состояние работы бензовозов. В конце месяца результат собирается в отдельный файл Excel, которые содержат данные по всем АТЗ за месяц (Рисунок 32). При формировании отчетов в рамках бизнес–процесса сразу пишутся служебные записки по фактам несоответствия данных ГСМ и передаются в отдел службы безопасности холдинга. В сформированном отчете представлены следующие данные:

- место заправки;
- дата и время начала заправки;
- дата и время конца заправки;
- объем заправки по проточному датчику из ПО «АвтоГРАФ»;
- объем заправки по ведомости;
- разница в объемах;
- длительность заправки;
- водитель заправленной техники.

Список необходимых операций для формирования отчетности с периодичностью 2–3 дня, формирования отчета за месяц, а также время на составление служебных записок указано в таблице 16. На данный момент в холдинге функционирует и постоянно передает данные 32 бензовоза. В среднем формируется по 9 отчетов на каждый бензовоз и десятый общий за весь месяц.

Отчет о работе топливозаправщика Урал борт.№604, на участке Караган с 1.10.2018 по 31.10.2018

Местоположение	Время		ЗАПРАВКА АТЗ		Разница ДУТ/Вед.-сть	ЗАПРАВКИ ТС		Разница ПОРТ/Вед.-сть	Продолжительность	Водитель/ТС
	Начало	Конец	ДУТ, Л	Ведомость, л		ПОРТ-3, Л	Ведомость, л			
Объем цистерны топливозаправщика: 7100 л. Начальный уровень топлива в цистерне: 3545,5 л.										
Караган	01.10.2018 7:41:46	01.10.2018 7:46:37				691,90	692	-0,10	00:04:51	Мельников А.А., Машинист
Караган	01.10.2018 7:56:07	01.10.2018 7:57:57				203,40	204	-0,60	00:01:50	0000 0064 6011
Караган	01.10.2018 7:59:47	01.10.2018 8:01:27				156,40	157	-0,60	00:01:40	0000 00C0 4166
Караган	01.10.2018 8:18:33	01.10.2018 8:20:50				257,20	257	0,20	00:02:17	Мельников В.И.,
Караган	01.10.2018 8:27:20	01.10.2018 8:29:32				263,80	264	-0,20	00:02:12	Корольков Г.В.,
Караган	01.10.2018 8:31:01	01.10.2018 8:32:31				170,00	170	0,00	00:01:30	Задорожный А.А., Машинист
Караган	01.10.2018 8:33:32	01.10.2018 8:36:08				277,00	277	0,00	00:02:36	Степанюк И.А., водитель
Караган	01.10.2018 8:42:55	01.10.2018 8:44:39				167,90	168	-0,10	00:01:44	Фролов А.В., Машинист
Караган	01.10.2018 8:50:32	01.10.2018 8:51:55				157,40	157	0,40	00:01:23	Напрюшкин В.А.,
Караган	01.10.2018 8:52:59	01.10.2018 8:55:24				328,80	329	-0,20	00:02:25	Стефеенко В.Е., Машинист
Караган	01.10.2018 8:57:44	01.10.2018 9:01:14				566,60	567	-0,40	00:03:30	Кравченко С.П.,
Караган	01.10.2018	01.10.2018				268,60	269	-0,40	00:02:05	0000 0064 6011
Караган	01.10.2018	01.10.2018				218,00	218	0,00	00:01:48	0000 00C1 0F35
Караган	01.10.2018	01.10.2018				290,40	291	-0,60	00:02:25	Калиничев Р.Г.,
Караган	01.10.2018	01.10.2018				286,60	286	0,60	00:02:08	0000 0063 7D35
Караган	01.10.2018	01.10.2018				137,60	138	-0,40	00:01:21	Больдт И.В., Машинист
Караган	01.10.2018	01.10.2018				169,60	170	-0,40	00:01:46	Токмашов В.В., Машинист
Караган	01.10.2018	01.10.2018				518,60	519	-0,40	00:03:16	Кравченко С.П.,
Караган	01.10.2018	01.10.2018				150,60	152	-1,40	00:01:22	Винокуров Р.С., Водитель
Караган	02.10.2018 7:05:50	02.10.2018 7:07:22				76,10	76	0,10	00:01:32	Кравченко С.П.,
Караган	02.10.2018 7:10:16	02.10.2018 7:13:32				438,40	439	-0,60	00:03:16	Булатов В.И.,
Караган	02.10.2018 7:15:52	02.10.2018 7:18:30				382,40	383	-0,60	00:02:38	Малышев О.А., Машинист
Караган	02.10.2018 7:27:52	02.10.2018 7:30:07				334,80	335	-0,20	00:02:15	Кравченко С.П.,
Караган	02.10.2018 7:34:41	02.10.2018 7:36:23				142,30	143	-0,70	00:01:42	0000 00C1 131B
Караган	02.10.2018 7:36:53	02.10.2018 7:38:57				251,20	252	-0,80	00:02:04	0000 00C0 40D2
Караган	02.10.2018 7:39:25	02.10.2018 7:41:13				220,20	221	-0,80	00:01:48	0000 0064 6011
Караган	02.10.2018 7:47:50	02.10.2018 7:49:01				107,60	108	-0,40	00:01:11	Напрюшкин В.А.,
Караган	02.10.2018 8:06:12	02.10.2018 8:08:12				230,60	231	-0,40	00:02:00	Задорожный А.А., Машинист
Караган	02.10.2018 8:16:37	02.10.2018 8:18:29				164,20	165	-0,80	00:01:52	Фролов А.В., Машинист
Караган	02.10.2018 8:24:28	02.10.2018 8:26:28				241,20	242	-0,80	00:02:00	Стефеенко В.Е., Машинист
Караган	02.10.2018 8:29:46	02.10.2018 8:32:58				491,20	491	0,20	00:03:12	Кравченко С.П.,
Караган	02.10.2018 8:40:29	02.10.2018 8:44:01				525,20	526	-0,80	00:03:32	Кравченко С.П.,
Караган	02.10.2018 8:45:13	02.10.2018 8:47:21				176,20	176	0,20	00:02:08	Разбираев А.П.,
Караган	02.10.2018	02.10.2018				113,80	113	0,80	00:01:41	Разбираев А.П.,
Караган	02.10.2018	02.10.2018				236,30	238	-1,70	00:01:53	0000 00C0 4166
Караган	02.10.2018	02.10.2018				69,40	69	0,40	00:01:20	Кравченко С.П.,
Караган	02.10.2018	02.10.2018				321,80	322	-0,20	00:02:28	0000 00B6 4097

Рисунок 32 – Отчет о заправках техники АТЗ

Таблица 16 – Список заданий по формированию отчета о заправках техники

Сотрудник	Задания	Количество секунд на задание	Количество повторов задания	Время выполнения повторов задания в секундах
Старший аналитик	Проверить служебную записку	640	30	19200
	Проверить месячный отчет	490	21	10290
	Проверить трехдневный отчет	180	288	51840
Итого использовано времени старшего аналитика на отчет в секундах:				81330
Аналитик	Запросить данные в виде отчета за 3 дня	180	288	51840
	Выделить цветом ячейки по дням	15	288	4320
	Найти ведомость по АТЗ.	10	288	2880

Продолжение таблицы 16 – Список заданий по формированию отчета о заправках техники

Сотрудник	Задания	Количество секунд на задание	Количество повторов задания	Время выполнения повторов задания в секундах
Аналитик	Заполнить трехдневный отчет данными из ведомостей	180	288	51840
	Добавить в сводную таблицу новые данные	30	288	8640
	Написать служебную записку	1800	30	54000
	Перенести данные из трехдневных отчетов в месячный отчет	300	21	6300
Итого использовано времени аналитика на отчет в секундах:				179820
Всего использовано времени на отчет в секундах:				261150

Бизнес–процесс формирования отчета по заправкам техники АТЗ описан на рисунке 33.

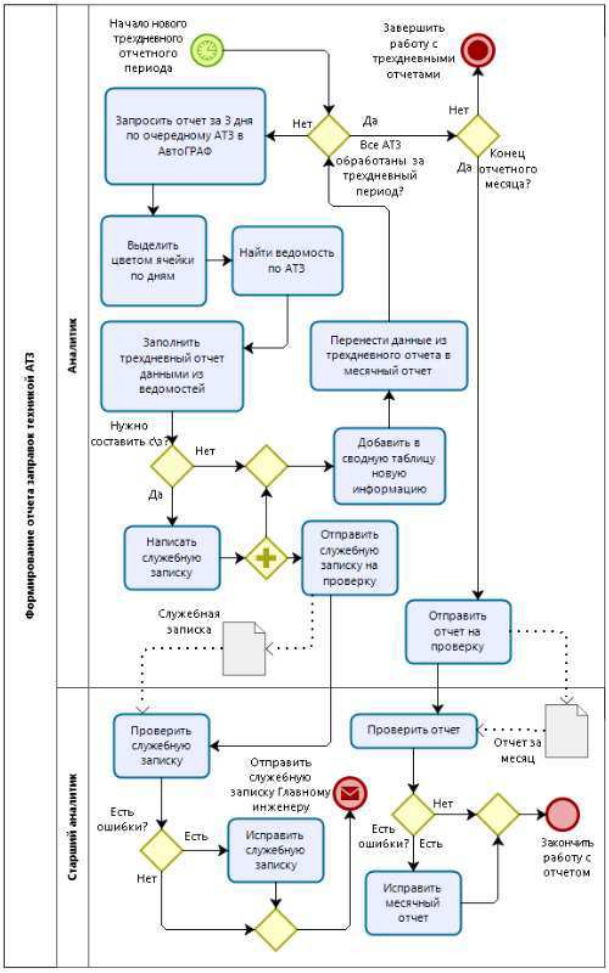


Рисунок 33 – Бизнес–процесс формирования отчетности по заправкам техники АТЗ

По данным из таблицы 16 аналитик формирует отчет за 49 часов и 57 минут (50 часов). Старший аналитик также участвует в процессе и тратит 22 часа, 35 минут и 30 секунд на него (22,5 часов). Суммарно полученное значение больше указанного в таблице 9 на 1,5 часов.

2.5 Обоснование необходимости разработки информационно–аналитической системы формирования отчетности

Описанные бизнес–процессы возможно изменить, автоматизировав ряд задач, которые сейчас выполняют аналитики. Для этого необходимо разработать и внедрить информационно–аналитическую систему формирования отчетности на предприятии. Использование ИАС позволит убрать из формирования отчетности большинство механических действий, на которые сейчас тратится рабочее время специалистов.

Для обоснования возможности увеличения эффективности отдела информационной аналитики и достижения конечной цели, указанной на рисунке 12, необходимо разработать новые бизнес–процессы с предлагаемой ИАС. Новые бизнес–процессы позволяют наглядно показать, как изменения положительно скажутся на работе отдела информационной аналитики.

С помощью описанного бизнес–процесса из пункта 2.4.1 и списка задач из таблицы 14 можно выделить задачи, которые можно автоматизировать в отчете по подрядчикам:

- создание нового отчета;
- проверка новых данных;
- выбор данных по сменам;
- перенос моточасов в отчет.

Предлагаемый бизнес–процесс показан на рисунке 34.

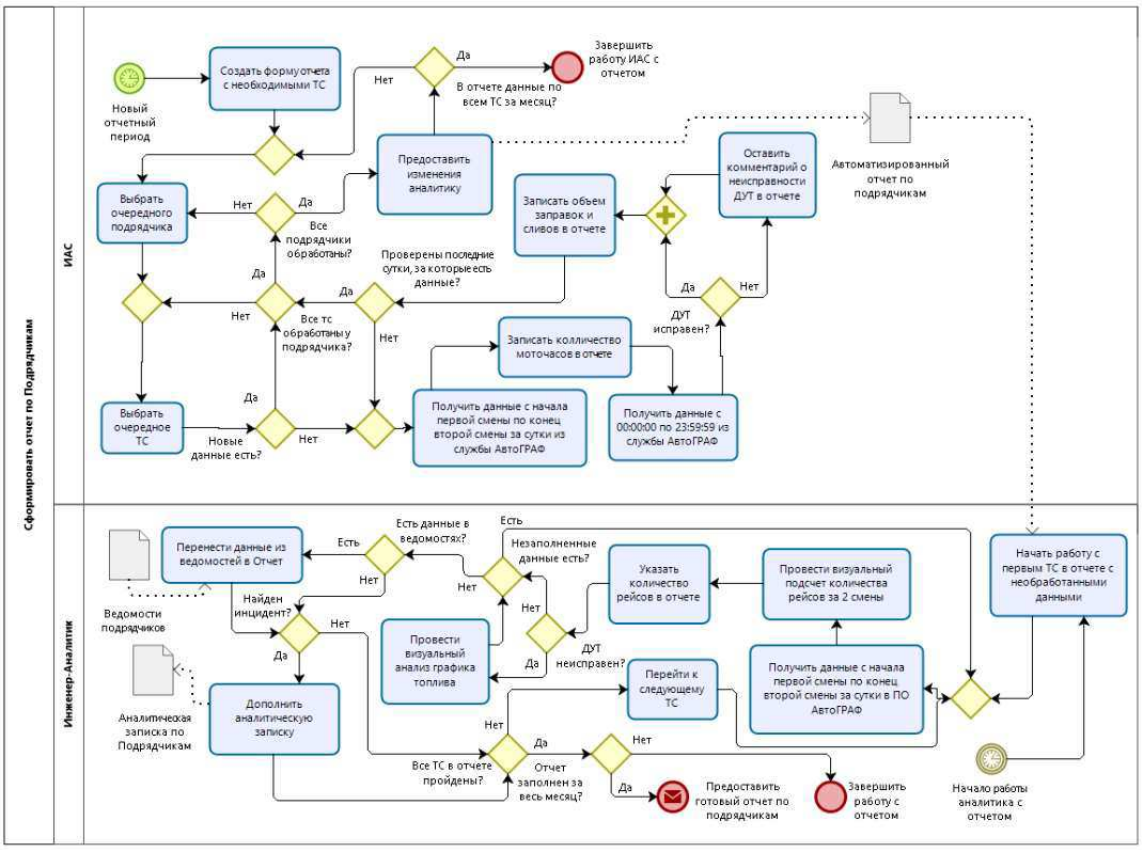


Рисунок 34 – Обновленный бизнес–процесс формирования отчета по подрядчикам

При использовании этого бизнес–процесса появляется возможность сократить время создания отчета на 11%.

С помощью описанного бизнес–процесса из пункта 2.4.2 и списка задач из таблицы 15 можно выделить задачи, которые можно автоматизировать в отчете по колесной и тяжелой технике холдинга:

- проверка наличия новых данных;
- запросы данных через диспетчерское ПО;
- запись заправок и сливов топлива в отчет;
- запись моточасов в отчет;
- продление листа техники для будущей отчетной недели.

Предлагаемый бизнес–процесс показан на рисунке 35.

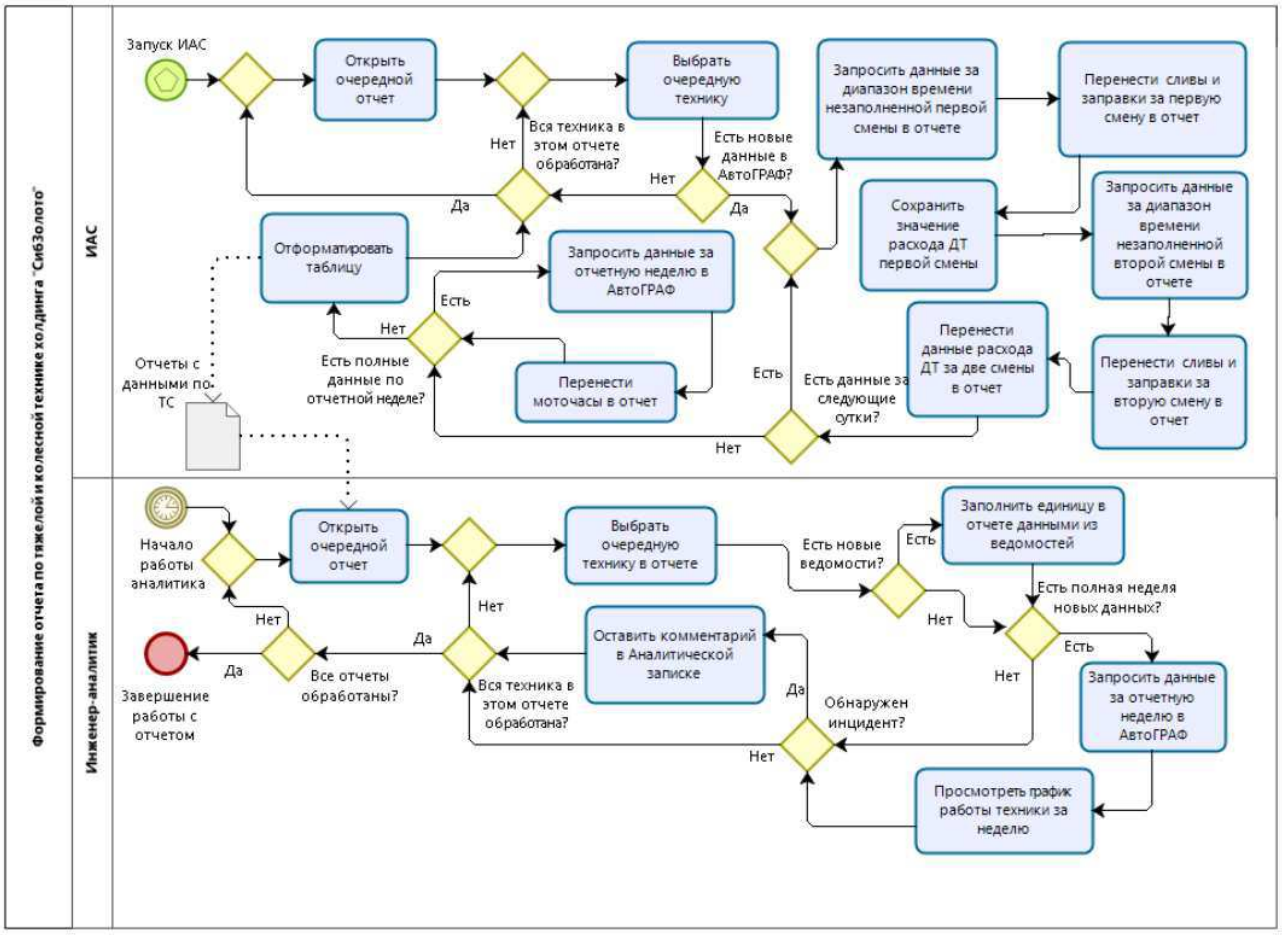


Рисунок 35 – Обновленный бизнес–процесс формирования отчета по колесной и тяжелой технике

После внедрения ИАС используемое рабочее время аналитиков на формирование данного отчета уменьшится на 69%.

С помощью описанного бизнес–процесса из пункта 2.4.3 и списка задач из таблицы 16 можно выделить задачи, которые можно автоматизировать в отчете по заправкам техники АТЗ:

- запрос отчета за три для по АТЗ;
- выделение цветом ячейки в таблице;
- перенести данные из трехдневных отчетов в месячный.

Также данный отчет предлагается вести накопительным методом, как отчет по колесной и тяжелой технике. Предлагаемый бизнес–процесс показан на рисунке 37.

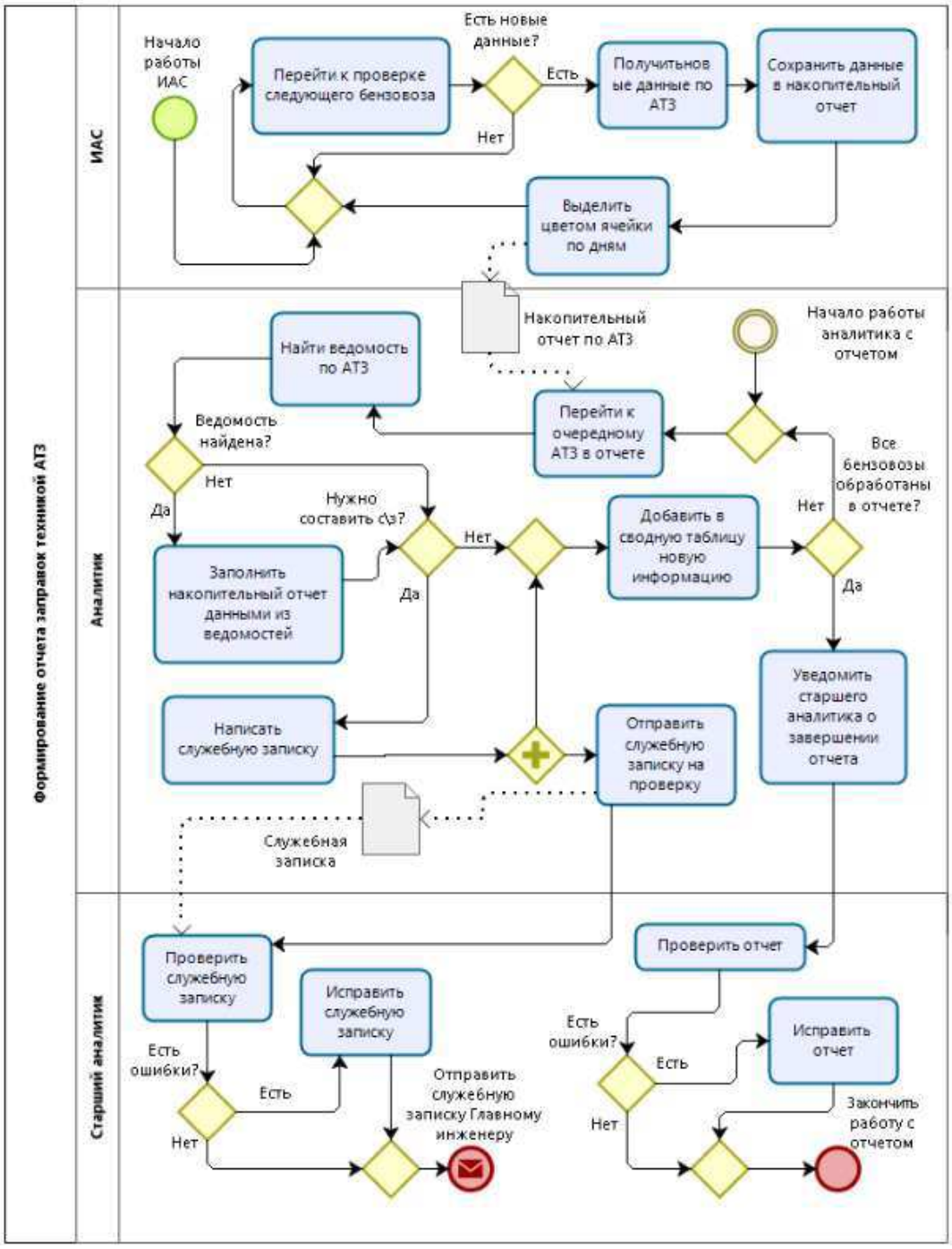


Рисунок 36 – Обновленный бизнес–процесс формирования отчета по заправкам техники АТЗ

После внедрения ИАС используемое время аналитика на формирование данной отчетности сократится на 35%, всего используемого времени на формирования отчетности уменьшится на 24%.

Рассматривая предложенные бизнес–процессы, можно сделать вывод, что разработка и внедрение информационно–аналитической системы формирования отчетности позволит освободить рабочее время специалистов, тем самым появиться возможность контролировать больший парк техники тем же составом сотрудников в отделе.

3 Разработка информационно–аналитической системы формирования отчетности для отдела информационной аналитики

3.1 Выбор средств разработки информационно–аналитической системы

При выборе средств разработки ИАС необходимо учитывать следующие ограничения:

− новый программный продукт должен быть максимально похож на старую среду работы с отчетами, чтобы внедрение изменений прошло безболезненно;

− необходимо использовать преимущественно бесплатные решения, так как бюджет на разработку ИАС отсутствует.

Рассматривалось три варианта реализации системы:

1) Отдел информационной аналитики продолжает использовать Excel для формирования отчетности и обработки данных по технике. Автоматически созданные отчеты будут храниться на FTP–сервере, с которого аналитики будут их скачивать себе на рабочий компьютер и работать с ними.

2) Разработать новый модуль в панели мониторинга, в котором будут располагаться сформированные отчеты. Сотрудники будут работать с данными в панели мониторинга через web–интерфейс, в котором необходимо будет реализовать рабочую среду, схожую по функциональным возможностям и внешнему интерфейсу с Excel.

Формировать отчетность на файловом хостинге Google Drive, используя их интегрированный бесплатный пакет офисных web–приложений (Google Docs). Отдел ИА будет работать с отчетами в среде Google Spreadsheets.

В таблице 17 указаны функциональные требования к системе и процессу разработки, и возможности каждого решения.

Таблица 17 – Сопоставление возможностей вариантов реализации с критериями к разрабатываемой системе

Критерий	Excel\FTP–сервер	Панель мониторинга	Google Spreadsheets
Стоимость разработки	Время отдела РСПО	Дополнительные выплаты разработчикам панели	Время отдела РСПО
Ожидаемое время разработки	3–4 месяца	По оценке ТЗ разработчиками, и полгода технического долга по разработке	1–2 месяца
Сложность адаптации аналитиков к новой рабочей среде	Нулевая сложность	В зависимости от реализации	Около одной рабочей недели
Режим доступа к отчетности	наличие доступа к FTP–серверу	наличие доступа к панели мониторинга	Разрешение на просмотр и изменения в Google Drive

Окончание таблицы 17 – Сопоставление возможностей вариантов реализации с критериями к разрабатываемой системе

Критерий	Excel\FTP–сервер	Панель мониторинга	Google Spreadsheets.
Возможность совместной работы по отчету несколькими сотрудниками одновременно	В последней версии Excel есть возможность совместной работы	Невозможно	Есть возможность
Online–добавление данных при работе с отчетом	Возможно	Возможно	Возможно
История изменений отчета сотрудниками	В последней версии Excel есть возможность просмотра изменений пользователей	Можно добавить такой функционал	Есть возможность
Расположение отчетов	В локальной сети холдинга	в локальной сети холдинга	на внешнем хостинге Google Drive
Доступ к отчетам из офиса	Без ограничений	Без ограничений	Не будет, при отсутствии интернета
Доступ вне офиса	При наличии доступа на FTP–сервер	При наличии интернета	При наличии интернета

В первом варианте главными преимуществами являются использование внутренних ресурсов компании на реализацию системы и возможность пользователям остаться в привычной среде Excel. Минусами являются отсутствие важных функциональных возможностей, указанных в требованиях, отсутствие последних версии программы на предприятии и большое количество затрат времени отдела РСПО на разработку ИАС.

Во втором варианте главным преимуществом является разработка в существующей системе, в которой уже есть инструменты получения данных по технике. Минусами являются затраты на реализацию, срок реализации, а также отсутствие гарантий удобства работы для аналитиков.

В третьем варианте преимуществами являются использование внутренних ресурсов компании для реализации ИАС, короткий срок внедрения программного продукта, а также наличие свойств, заявленных в требованиях. Недостатком решения является невозможность работать с ним при отсутствии интернета в офисе.

Вариант реализации через Google Docs имеет больше преимуществ.

После выбора среды, где будет формироваться отчетность, необходимо определить язык программирования (ЯП), на котором будет написана логика формирования отчетности. В таблице 18 приведено сравнение популярных языков программирования и критерии, которые учитываются при выборе ЯП.

Таблица 18 – Сопоставление популярных языков программирования по критериям к разработке ИАС

Характеристика	C#	JavaScript	Python
Сложность разработки	Высокая	Средняя	Низкая
Скорость разработки	Низкая	Средняя	Высокая
Использование ресурсов сервера	Низкая	Высокая	Средняя
Что необходимо реализовать	Необходимо реализовать взаимодействие с Google API, AutoGRAPH.NET и бизнес–логику формирования отчетности	Необходимо реализовать взаимодействие с Google API, AutoGRAPH.NET и бизнес–логику формирования отчетности	Уже имеется код для работы AutoGRAPH.NET, необходимо реализовать взаимодействие с Google API и бизнес–логику отчетов
Создание автономного приложение	Имеются встроенные модули для создания приложения как службы Windows	Необходимо разворачивать node.js для использования JS на сервере	Есть возможность использовать подключаемые модули для создания приложения как службы Windows
Поддержка Google API	Есть возможность	Есть возможность	Есть возможность
Многопоточность	Имеется	Отсутствует	Имеется в бета версии языка

На всех этих языках уже есть библиотеки для работы с Google API.

При реализации сервиса на C# преимуществами являются выполнение в среде .NET, которое имеет встроенные ресурсы оптимизации использования ресурсов, возможность реализовать многопоточное приложение, и простая реализация создания программы как службы Windows. Недостатками данного языка являются сравнительно высокая сложность и скорость разработки относительно других вариантов.

При реализации сервиса на JavaScript явных преимуществ не наблюдается. Минусами являются сложное развертывание решения на локальном сервере, сравнительно большое использование ресурсов локального сервера и отсутствие многопоточности.

При реализации сервиса на Python преимуществами являются низкая сложность разработки в купе с высокой скоростью реализации языка, наличием уже написанного кода, взаимодействующего с AutoGRAPH.NET, что также

ускоряет время разработки, и дает возможность с помощью дополнительных модулей языка реализовать решение как службу Windows. Минусами является наличие многопоточности только в бета–версии языка.

Общая схема работы ИАС показана на рисунке 37.

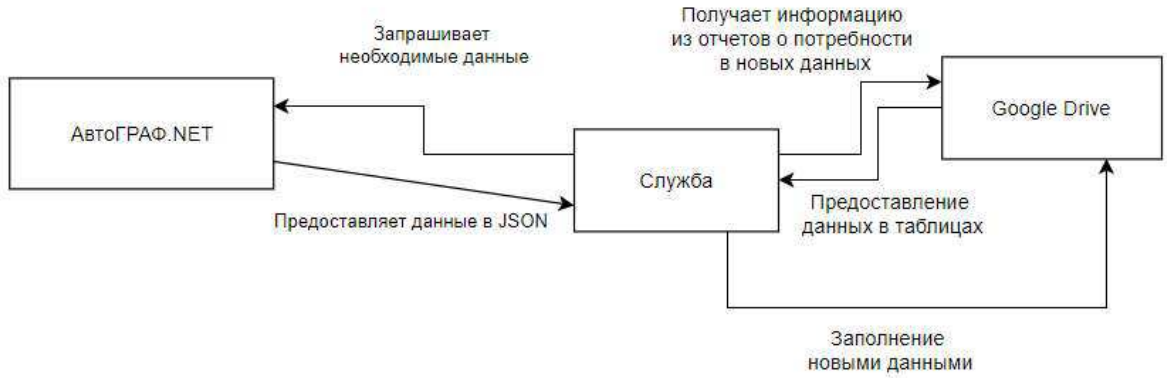


Рисунок 37 – Схема работы информационно–аналитической системы

Лучшим вариантом является Python, так как он имеет больше преимуществ перед другими ЯП. Кроме того, многопоточность для информационно–аналитической системы не востребована, по трем причинам:

1) Нет потребности в высокой скорости обновления данных в отчетах. Достаточно своевременной дозагрузки данных при их появлении.
2) Наличие квот, которые создают верхний порог производительности при взаимодействии с Google API.
3) AutoGRAPH.NET также ограничен в ресурсах, и уже взаимодействует с панелью мониторинга, увеличение количества параллельных запросов приведет к некорректной работе панели.

3.2 Описание процесса разработки информационно–аналитической системы формирования отчетности

Весь процесс разработки и внедрения ИАС поэтапно можно разделить следующим образом:

1) Разработка прототипа автоматического отчета в Google Docs (Заправки техники АТЗ).
2) Создание класса для работы с Google API.
3) Разработка процесса формирования отчета по заправкам техники АТЗ.
4) Предоставление отчета аналитикам и исправление найденных ошибок и замечаний.
5) Разработка процесса формирования отчета по колесной и тяжелой технике холдинга.
6) Предоставление отчета аналитикам и исправление найденных ошибок и замечаний.

7) Перенос аналитических записок по подрядчикам и по колесной и тяжелой технике холдинга на Google Drive.

8) Разработка процесса формирования отчета по подрядчикам.

9) Предоставление отчета аналитикам и исправление найденных ошибок и замечаний.

10) Перевод системы на автономную работу.

3.2.1 Прототип автоматического отчета в Google Spreadsheets

Так как Google Spreadsheets отличается от Excel, внедрение таких изменений может негативно отразиться на работе отдела. Поэтому первым этапом разработки стало создание прототипа отчета во внутренней среде Google – использовался язык Google Apps Script, JavaScript-подобный язык, который реализован во внутренней архитектуре облачных систем Google. Результатом работы данного отчета являлась оперативная выгрузка данных по заправкам техники АТЗ. Формат и оформление таблицы задается автоматически.

Код прототипа показан в приложении А. В нем реализованы две основные функции:

– обновление списка опрашиваемых бензовозов;

– занесение данных по заправкам.

Запуск скрипта, формирующего отчет реализован через верхнее меню таблицы с отчетом, дополнительным пунктом (Рисунок 38).

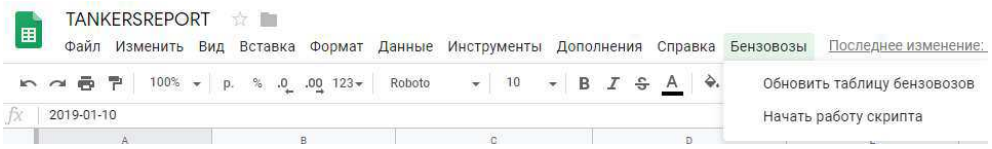


Рисунок 38 – секция в верхнем меню для прототипа

Для работы с AutoGRAPH.NET была изучена документация по работе с ним. [19]

Для решаемых задач в данном отчете необходимо использовать 3 метода:

– GetOnlineInfoAll, возвращает последние актуальные данные по всей технике;

– EnumDevices, возвращает список всех единиц в схеме;

– GetStage, возвращает отрезки по определенному параметру указанного объекта мониторинга за выбранный период времени.

Метод EnumDevices требуется для получения ID объекта в схеме. Только этот метод дает возможность определить технику по бортовому номеру или номера модуля Автограф.

Метод GetOnlineInfoAll требуется для получения даты и времени последних данных у объекта, чтобы узнать, пришли ли новые данные, и до какой даты целесообразно делать запрос.

Метод GetStage требуется для получения значений, которые необходимо занести в отчете.

В случае с обновлением списка опрашиваемых бензовозов, прототип забирал данные по работающим АТЗ, который ведет главный инженер в сводной таблице бензовозов (на момент создания прототипа была перенесена в Google Drive). В сводной таблице требовалась следующая информация:

- имя бензовоза;
- гос.номер;
- участок;
- номер установленного на него Автографа.

После получения данных из сводной таблицы, прототип проводил соответствие между номерами Автографа и ID транспортного средства с помощью запроса метода EnumDevices. После этого сформированный список бензовозов сохранялся в первом листе таблицы. На первом листе таблицы также указывалась дата последних данных в отчете и результат последней обработки бензовоза.

Логика обработки бензовозов последовательна. При запуске программы происходило считывание всех данных с первого листа, каждая строка которой – бензовоз с информацией о нем для корректного заполнения отчета. Следующий шаг – получение JSON объекта метода GetOnlineInfoAll, в котором хранится время последних данных по каждой техники. При условии разницы в датах из листа со всеми бензовозами и датой из запроса GetOnlineInfoAll и наличии уже существующего листа с данным бензовозом (определение листа с бензовозом исходило из самого названия листа, которые назывались как «Имя_Бензовоза Номер_Автограф», для создания уникальных значений), запрашивался метод GetStage с необходимыми параметрами, с даты общего листа, с 00:00:00 по дату из GetOnlineInfoAll, со временем 00:00:00. После получение JSON–объекта происходил перенос данных в таблицу – на листе обрабатываемого бензовоза программа получала последнюю заполненную строку, относительно нее происходило добавление новых данных. После этого на общем листе менялась дата последних данных, а в статусе обработки бензовоза на общем листе указывалось, что обработка прошла успешно. Если же различие в датах имелось, но листа с бензовозом не существовало в таблице, создавался лист, форматировалась шапка листа с данными по бензовозу, а после этого происходила работа алгоритма, как если бы лист уже был бы создан.

Если при работе с бензовозом происходила какая–то ошибка, информация о ней также указывалась в результате обработки, и код переходил к следующей единице.

После реализации данного скрипта ссылка на данный отчет была предоставлена в отдел информационной аналитики для тестирования и знакомства с функциональными особенностями Google–таблиц. Листинг кода прототипа в приложении А.

Сотрудники освоились с работой в новой среде, однако вскоре были выявлены недостатки данной версии прототипа.

В первую очередь, ограничения Google не позволяли работать скрипту более 6 минут. При запуске по таймеру нет возможности работать более, чем на полчаса в сутки, или не более 5 раз в сутки. [20] В соответствии с этим, возникали проблемы с оперативным заполнением данных в таблице, даже при полном использовании временных квот прототип ИАС не всегда обрабатывал все бензовозы из общего листа. Это было связано с тем, что AutoGRAPH.NET мог продолжительное время обрабатывать запрос GetStage. Функция Google App Script UrlFetchApp.fetch(url), которая выполняла Get–запрос, возвращала ошибку, если не получала ответ за минуту. В половине случаев AutoGRAPH.NET возвращал данные в приемлемый срок, в остальных случаях алгоритм работы доходил по времени до критического значения в одну минуту и переходил дальше по циклу. Долгий ответ от AutoGRAPH.NET происходил по двум причинам:

- временной промежуток между последними данными в таблице и последними данными на сервере мог быть более нескольких недель, и из–за большого объема данных формирование JSON–объекта требовало времени;
- некоторые АТЗ имели большой объем данных за короткий промежуток времени. Связано это было непосредственно с поломкой оборудования на бензовозе.

При этом, с данной версией аналитики стали работать быстрее и положительно оценили новую среду.

С помощью прототипа был получен следующий результат:

- появилось однозначное понимание технологии работы с AutoGRAPH.NET;
- решение опрашивать единицы, которые указаны вручную в другой таблице, несет риски: обновление справочной таблицы может быть недостаточно оперативным, человеческий фактор, опечатки в значениях, которые не предусмотрены в программе, увеличение количества сущностей, которые необходимо актуализировать.

Первый пункт дал точное понимание структуры ответов сервиса на основные запросы, а также показал проблему долгого формирования ответа при запросе данных за большой промежуток времени.

Последний пункт внес коррективы в формирование списка техники для определенных видов отчетов. Было решено добавить для каждого объекта мониторинга в схеме новый атрибут – тип техники. Указывались следующие типы техники:

- HEAVYEQ – тяжелая техника;
- WHEELVEH – колесная техника;
- GASOLINE – бензовозы;
- STATOBJ – стационарный объект (ДЭС или ДНС);
- GASSTATION – склад ГСМ.

3.2.2 Создание методов для работы с Google API

После внедрения прототипа началась разработка информационно–аналитической системы. Для удобства разработки использовался Git – распределенная система контроля версии [21]. Хостингом послужил ресурс Gitlab, там уже располагались репозитории с кодом диспетчерской панели.

Для логгирования работы ИАС была выбрана библиотека «logging» [22]. Используемые функции библиотеки:

- указание формата строки лога;
- понимание библиотекой, в каком файле выполняется код;
- наличие различных уровней логгирования.

Для решения задач отдела РСПО ранее был написан код на Python, позволяющий взаимодействовать с AutoGRAPH.NET (Приложение Б).

С помощью документации [23] по работе с Google API через ЯП Python, была организована среда разработки. Скачаны библиотеки:

- httplib2;
- googleapiclient;
- oauth2client.

Для работы с Google API необходимо создать сервисный аккаунт, создать проект и сгенерировать закрытый ключ, который необходим для работы с Google API (Рисунок 39).

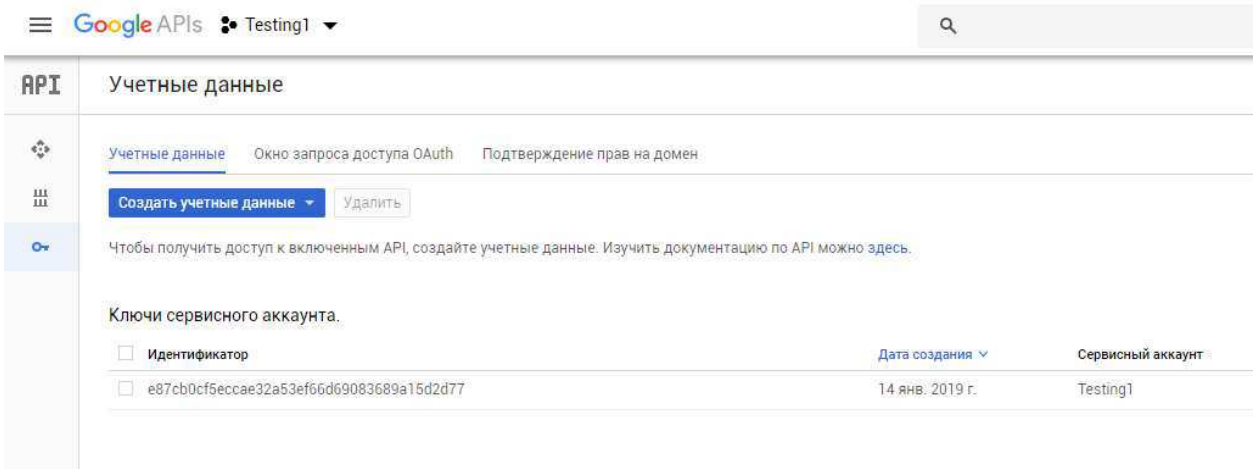


Рисунок 39 – Создание сервисного аккаунта

После этого необходимо добавить сгенерированный ключ в формате json к себе в проект и ссылаться на него при подключении к Google API.

Для большего удобства разработки информационно–аналитической системы был написан собственный класс GoogleSheetsAPI, методы которого позволяли ускорить разработку ИАС. Листинг кода в приложении В.

Во время тестирования и написания собственных функций к Google Spreadsheet API было выявлено, что, в соответствии с квотами на использование Google Spreadsheets API, нельзя превышать лимит 500 запросов в 100 секунд. Соответственно вся дальнейшая разработка велась с учетом этого ограничения.

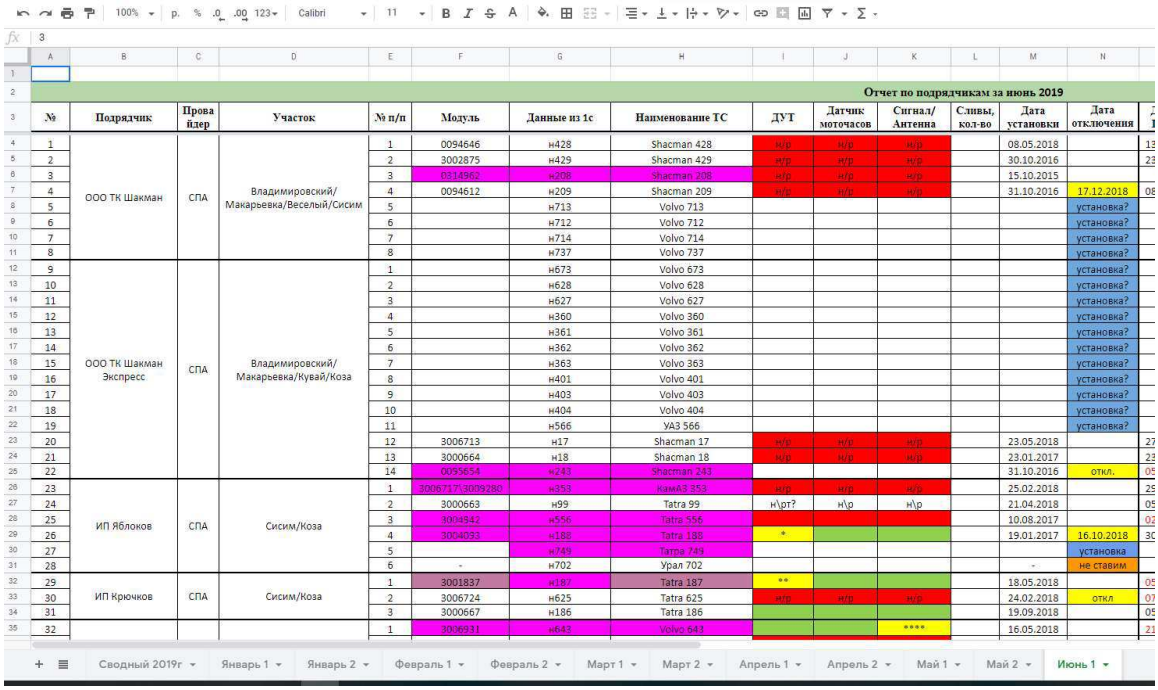
3.2.3 Описание алгоритма работы информационно–аналитической системы

Рассмотрим логику работы при очередном запуске программы.

В первую очередь, при запуске программы происходит создание экземпляров класса Reports() (в методах класса описана бизнес–логика формирования отчетности), NetServiceUrl() (методы класса позволяют взаимодействовать с AutoGRAPH.NET) и GoogleSheetsAPI() (методы класса позволяют взаимодействовать с Google API). Далее вызывается основная функция Main(), с объектом логирования и ранее созданными экземплярами классов.

В основной функции сразу идет запрос GetOnlineInfoAll, чтобы получить информацию о последних данных по технике. После этого идет обновление данных в аналитической записке по подрядчикам с помощью метода R.UpdateActualDataConstractors, который в качестве аргумента принимает JSON–объект метода GetOnlineInfoAll.

Перед созданием автоматического отчета по подрядчикам была перенесена аналитическая записка по подрядчикам на Google Drive (Рисунок 40), по причине необходимости получать информацию из данной аналитической записки при формировании отчета по подрядчикам.



Отчет по подрядчикам за июнь 2019

№	Подрядчик	Провайдер	Участок	№ п/п	Модуль	Данные из 1с	Наименование ТС	ДУТ	Датчик моточасов	Сигнал/ Антенна	Сливы, кол-во	Дата установки	Дата отключения
1	ООО ТК Шакман	СПА	Владимировский/ Макарьевка/Веселый/Сисим	1	0094646	н428	Shacman 428	н/р	н/р	н/р		08.05.2018	
2				2	3002875	н429	Shacman 429	н/р	н/р	н/р		30.10.2016	
3				3	0314962	н208	Shacman 208	н/р	н/р	н/р		15.10.2015	
4				4	0094612	н209	Shacman 209	н/р	н/р	н/р		31.10.2016	17.12.2018
5				5		н713	Volvo 713						установка?
6				6		н712	Volvo 712						установка?
7				7		н714	Volvo 714						установка?
8				8		н737	Volvo 737						установка?
9	ООО ТК Шакман Экспресс	СПА	Владимировский/ Макарьевка/Кувай/Коза	1		н673	Volvo 673						установка?
10				2		н628	Volvo 628						установка?
11				3		н627	Volvo 627						установка?
12				4		н360	Volvo 360						установка?
13				5		н361	Volvo 361						установка?
14				6		н362	Volvo 362						установка?
15				7		н363	Volvo 363						установка?
16				8		н401	Volvo 401						установка?
17				9		н403	Volvo 403						установка?
18				10		н404	Volvo 404						установка?
19				11		н566	УАЗ 566						установка?
20				12	3006713	н17	Shacman 17	н/р	н/р	н/р		23.05.2018	
21				13	3000664	н18	Shacman 18	н/р	н/р	н/р		23.01.2017	
22				14	0055654	н243	Shacman 243					31.10.2016	откл.
23	ИП Яблоков	СПА	Сисим/Коза	1	300671713009280	н353	Камаз 353	н/р	н/р	н/р		25.02.2018	
24				2	3000663	н99	Tatra 99	н\рт?	н\р	н\р		21.04.2018	
25				3	3004542	н556	Tatra 556					10.08.2017	
26				4	3004033	н188	Tatra 188	*				19.01.2017	16.10.2018
27				5		н749	Tatra 749						установка
28				6	-	н702	Урал 702					-	не ставим
29	ИП Крючков	СПА	Сисим/Коза	1	3001837	н187	Tatra 187	**				18.05.2018	
30				2	3006724	н625	Tatra 625	н/р	н/р	н/р		24.02.2018	откл
31				3	3000667	н186	Tatra 186					19.09.2018	
32				1	3006931	н643	Volvo 643			****		16.05.2018	

Рисунок 40 – Аналитическая записка по подрядчикам

В данной таблице создается лист с новым отчетным периодом при необходимости и обновляются даты последних данных по единицам техники подрядчиков.

После обновления аналитической записки по подрядчикам происходит запрос EnumDevices, JSON объект которого обрабатывается методом

Reports.GetVehicle(), в результате чего вся информация по технике преобразуется в список объектов, которые содержат следующие ключи:

- ID – ID объекта мониторинга в схеме;
- Name – имя объекта мониторинга из схемы;
- AreaID – ID участка, за которым закреплен объект;
- Area – название участка, за которым закреплен объект;
- CompanyID – ID направления, за которым закреплен объект;
- Company – название направления;
- Serial – бортовой номер Автографа;
- RegNumber – бортовой номер объекта;
- StateNumber – гос.номер объекта;
- Type – тип объекта.

После этого происходит работа с аналитической запиской по тяжелой и колесной технике (Рисунок 41). Проверяется, есть ли в аналитической записке лист с актуальным отчетным периодом, и, если он отсутствует, формирует лист со списком объектов из схемы, тип которого подходит под данную записку. За это отвечает метод R.UpdateAnalyticsTable().

fx Аналитическая записка работе тяжелой и колесной техники за период с 27.05.2019 по 02.06.2019

	A	B	C
1	**Аналитическая записка работе тяжелой и колесной техники за период с 27.05.2019 по 02.06.2019**		
2	**Участок**	**ТС**	**Комментарий**
3	**Север**		
4	Гаревая	Урал 375 "Бензовоз" №452	ок
5	Ильинский	Урал 375 "Бензовоз" №90067	Последняя передача данных была 29.03.19.
6	Ильинский-Петропавловский	Урал 375 "Бензовоз" №268	Последняя передача данных была 24.05.19.
7	Ильинский-Петропавловский	Урал 375 "Автоманипулятор" №270	Последняя передача данных была 25.05.19.
8	Ильинский-Петропавловский	МоАз 75054-222 "Самосвал" №1256	Последняя передача данных была 23.05.19.
9	Ильинский-Петропавловский	Погрузчик LW-500 KL №1245	Последняя передача данных была 21.03.19.
10	Ильинский-Петропавловский	Камаз 4318 "Бензовоз" №1297	Последняя передача данных была 13.05.19.
11	Ильинский-Петропавловский	МоАз 75054-222 "Самосвал" №2204	Последняя передача данных была 28.05.19.
12	Мамон	Д-85 №287	Последняя передача данных была 7.02.19.
13	Мамон	Урал 375Е "Грузовая цистерна" №170	Последняя передача данных была 31.05.19. Требуется тарировка ДУТ в собс
14	Мамон	Экскаватор Hyundai 430LC-9SH №299	Последняя передача данных была 28.05.19.
15	Озерный	Бульдозер D-355 4453 №210	Некорректно работает ДУТ.
16	Озерный	Урал 375 "Водовоз" №223	Последняя передача данных была 27.05.19. Некорректно работает ДУТ.
17	Озерный	Урал АТЗ 12-4320 №1229	ок
18	Озерный	Экскаватор LIEBHERR 944B HD-SL №1227	Последняя передача данных была 1.06.19.
19	Озерный	Бульдозер Komatsu D375A-5 №1254	Нестабильно работает ДУТ. Последняя передача данных была 30.05.19.
20	Озерный	Экскаватор LIEBHERR 944B HD-SL №1226	ок
21	Озерный	УАЗ №1268	Последняя передача данных была 18.05.19.
22	Озерный	Камаз №90084	Последняя передача данных была 15.02.19.
23	Озерный	Бульдозер D-355 №1240	Последняя передача данных была 26.05.19.
24	Озерный	Экскаватор LIEBHERR 944B HD-SL №1248	ок
25	Озерный	Экскаватор LIEBHERR 944B HD-SL №2287	ок
26	Озерный	НЕФАЗ 4208-03 "Вахта" №1291	ок
27	Озерный	Урал "Вахта" №2243	Нужно уточнить бортовой номер ТС.
28	Озерный	Бульдозер Komatsu Д275А-5 №90057	Последняя передача данных была 30.05.19.
29	Озерный	Урал АТЗ 5668Т5 №90085	Последняя передача данных была 11.05.19.
30	Озерный	Урал "Вахта" №1263	ок
31	Панимба	УАЗ 390945 "Фермер" №1255	Последняя передача данных была 12.03.19.
32	Панимба	Погрузчик LW500 №1206	Последняя передача данных была 22.05.19.
33	Панимба	Погрузчик LW-500 FL №1208	ок
34	Панимба	УАЗ №2223	Последняя передача данных была 1.06.19.

Рисунок 41 – Аналитическая записка по колесной и тяжелой технике

После обновления этих аналитических записок идет поэлементное прохождение каждого элемента списка объектов из схемы. В теле цикла идет проверка на наличие данного объекта в GetOnlineInfoAll, и если он есть в

запросе, происходит проверка наличие данных за актуальный год, и формируется определенный отчет (По заправкам техники АТЗ или тяжелой колесной). После обхода всех объектов в схеме происходит обновление отчета по подрядчикам, с помощью данных из аналитической записки. Перед описанием алгоритмов создания отчетов стоит заметить что в каждой таблице этих отчетов реализован лист с общим списком всех ТС, для более удобной работы.

В процессе внедрения данной системы основные таблицы со справочными данными были перенесены на Google Drive (Рисунок 42).

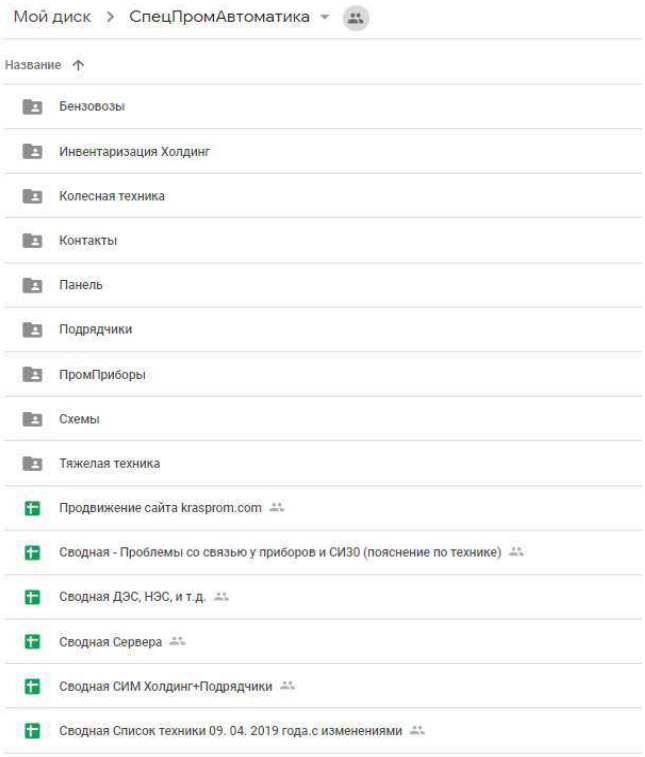


Рисунок 42 – Иерархия файлов и папок в Google Drive

В ходе разработки формирование уже созданных отчетов инициализировалось вручную запуском основного файла программы. После завершения основного этапа разработки был создан bat–файл, который запускается планировщиком заданий Windows (Рисунок 43) дважды в день.

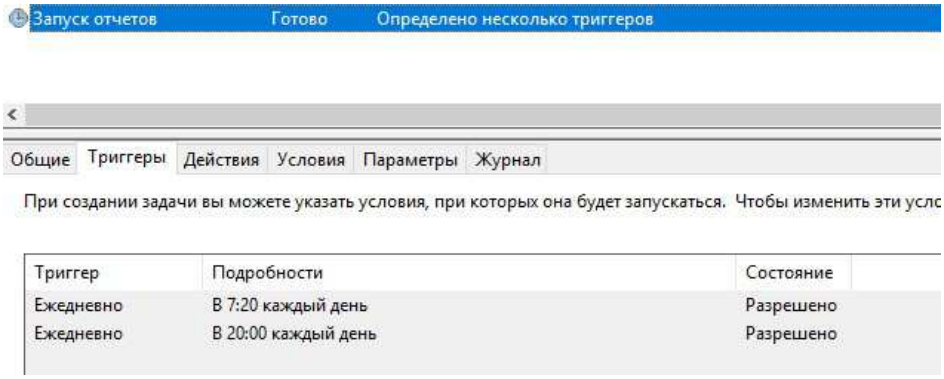


Рисунок 43 – Планировщик задач Windows

В среднем одна итерация запуска кода с утра после буднего дня занимает от одного до трех часов реального времени. Запуск кода после выходных выполняется чуть дольше, от полутора до четырех часов реального времени.

3.2.4 Описание бизнес–логики формирования отчета по заправкам техники АТЗ

В процессе разработки отчет по заправкам техники АТЗ был реализован первым, так как, в первую очередь, уже был написан прототип отчета на Google App Script, также работа в тестовом отчете уже велась параллельно со старым методом формирования отчетности, поэтому необходимо было исправить основные недочеты, а именно:

- получать бензовозы из схемы Автографа, а не сводной таблице по бензовозам;
- получать данные по всем бензовозам, с проблемными данными в том числе.

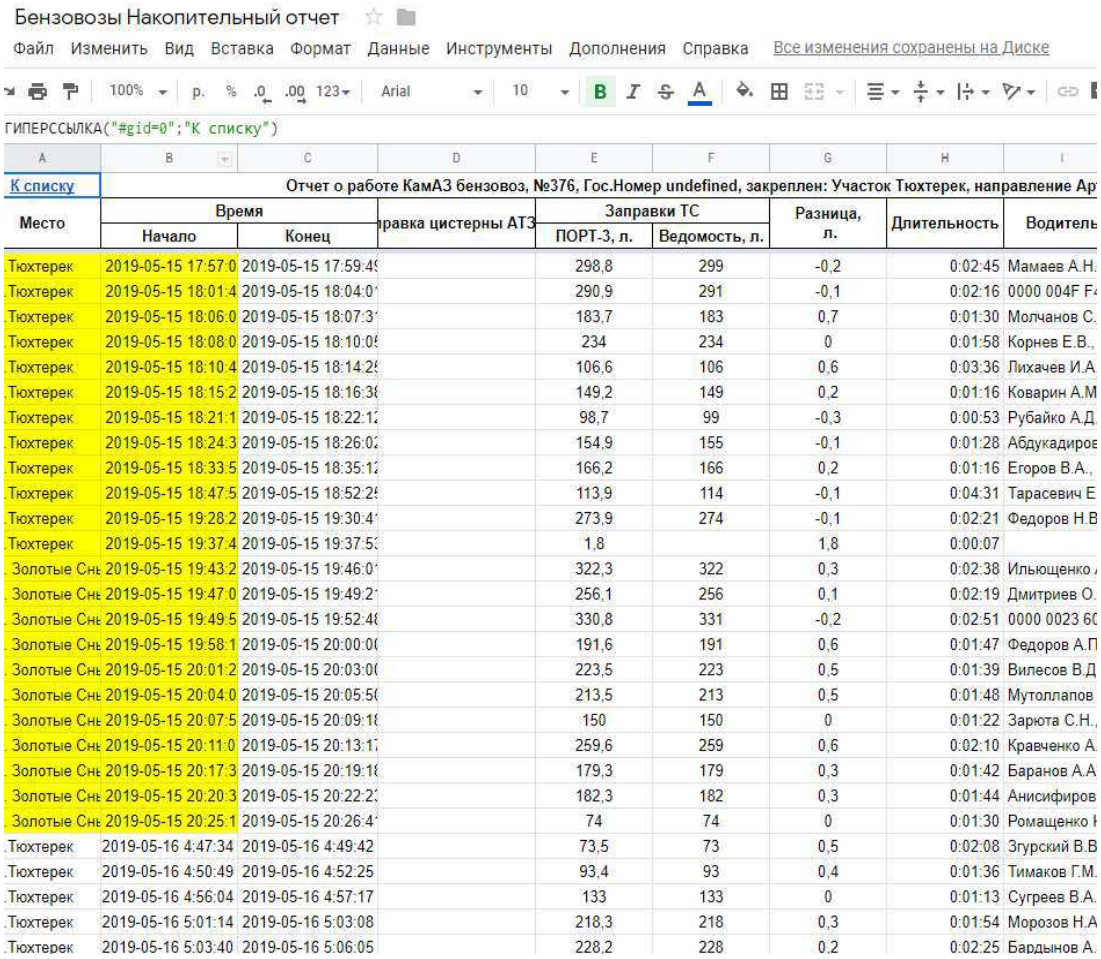
Работа алгоритма обработки данных схожа с первой версией. Основные изменения:

1) Имя листа теперь состоит из ‘'№{TC['RegNumber']} {TC['Name']} {TC['Serial']}'’;
   a. RegNumber – бортовой номер машины;
   b. Name – имя ТС, указанное в схеме Автографа;
   c. Serial – модуль автографа.
2) Ключ поиска нужного листа происходит только по RegNumber и Serial. Такое решение позволяет проще ориентироваться в листах таблице (с «именем» бензовоза), при этом уникальный ключ остается неизменным с прошлой версии. Инженеры–аналитики могут переименовать имя бензовоза, на работу отчета это не повлияет.

При отсутствии листа с данным бензовозом создается новый лист и оформляется шапка отчета и первичное форматирование (Объединение ячеек, полужирный шрифт, закрепленные столбы шапки). Далее алгоритм аналогичен, как если бы лист уже был создан ранее и найден при поиске.

Если лист найден, происходит получение номера последней строки на листе (отдельного метода в API Google нет для этого, реализация получения последней строки происходит через добавление в конец листа строки, закреплением возвращаемого индекса, удалением созданной строки). После этого получаются данные с последней строки, сравниваются с датой последних данных по ТС, и если между датами есть различие в сутки, запрашиваются данные за этот промежуток, как в первой версии отчета. Если записанный день по счету от начала года четный, заносим диапазон этого дня в отдельный диапазон, на основе которого будет проводиться цветовое разделение суток в отчете. При получении пустого ответа за сутки, это также указывается в отчете. На рисунке 44 показано, как выглядит данный вид отчета, с уже обработанными данными отделом информационной аналитики. Индекс

последней строки необходим для корректного написания формулы, которая в строке высчитывает разницу между данными из ведомости и данными системы мониторинга.


Бензовозы Накопительный отчет

Файл Изменить Вид Вставка Формат Данные Инструменты Дополнения Справка Все изменения сохранены на Диске

ГИПЕРССЫЛКА("#gid=0";"К списку")

	A	B	C	D	E	F	G	H	I
	К списку	Отчет о работе КамАЗ бензовоз, №376, Гос.Номер undefined, закреплен: Участок Тюхтерек, направление Ар							
	Место	Время		правка цистерны АТЗ	Заправки ТС		Разница, л.	Длительность	Водитель
		Начало	Конец		ПОРТ-3, л.	Ведомость, л.			
	Тюхтерек	2019-05-15 17:57:0	2019-05-15 17:59:49		298,8	299	-0,2	0:02:45	Мамаев А.Н.
	Тюхтерек	2019-05-15 18:01:4	2019-05-15 18:04:0		290,9	291	-0,1	0:02:16	0000 004F F4
	Тюхтерек	2019-05-15 18:06:0	2019-05-15 18:07:3		183,7	183	0,7	0:01:30	Молчанов С.
	Тюхтерек	2019-05-15 18:08:0	2019-05-15 18:10:05		234	234	0	0:01:58	Корнев Е.В.,
	Тюхтерек	2019-05-15 18:10:4	2019-05-15 18:14:25		106,6	106	0,6	0:03:36	Лихачев И.А.
	Тюхтерек	2019-05-15 18:15:2	2019-05-15 18:16:36		149,2	149	0,2	0:01:16	Коварин А.М
	Тюхтерек	2019-05-15 18:21:1	2019-05-15 18:22:12		98,7	99	-0,3	0:00:53	Рубайко А.Д.
	Тюхтерек	2019-05-15 18:24:3	2019-05-15 18:26:02		154,9	155	-0,1	0:01:28	Абдукадиров
	Тюхтерек	2019-05-15 18:33:5	2019-05-15 18:35:12		166,2	166	0,2	0:01:16	Егоров В.А.,
	Тюхтерек	2019-05-15 18:47:5	2019-05-15 18:52:25		113,9	114	-0,1	0:04:31	Тарасевич Е
	Тюхтерек	2019-05-15 19:28:2	2019-05-15 19:30:47		273,9	274	-0,1	0:02:21	Федоров Н.В
	Тюхтерек	2019-05-15 19:37:4	2019-05-15 19:37:53		1,8		1,8	0:00:07	
	Золотые Сны	2019-05-15 19:43:2	2019-05-15 19:46:01		322,3	322	0,3	0:02:38	Ильющенко А
	Золотые Сны	2019-05-15 19:47:0	2019-05-15 19:49:21		256,1	256	0,1	0:02:19	Дмитриев О.
	Золотые Сны	2019-05-15 19:49:5	2019-05-15 19:52:48		330,8	331	-0,2	0:02:51	0000 0023 60
	Золотые Сны	2019-05-15 19:58:1	2019-05-15 20:00:00		191,6	191	0,6	0:01:47	Федоров А.П
	Золотые Сны	2019-05-15 20:01:2	2019-05-15 20:03:00		223,5	223	0,5	0:01:39	Вилесов В.Д
	Золотые Сны	2019-05-15 20:04:0	2019-05-15 20:05:50		213,5	213	0,5	0:01:48	Мутоллапов
	Золотые Сны	2019-05-15 20:07:5	2019-05-15 20:09:10		150	150	0	0:01:22	Зарюта С.Н.,
	Золотые Сны	2019-05-15 20:11:0	2019-05-15 20:13:17		259,6	259	0,6	0:02:10	Кравченко А.
	Золотые Сны	2019-05-15 20:17:3	2019-05-15 20:19:16		179,3	179	0,3	0:01:42	Баранов А.А
	Золотые Сны	2019-05-15 20:20:3	2019-05-15 20:22:22		182,3	182	0,3	0:01:44	Анисифиров
	Золотые Сны	2019-05-15 20:25:1	2019-05-15 20:26:47		74	74	0	0:01:30	Ромащенко Н
	Тюхтерек	2019-05-16 4:47:34	2019-05-16 4:49:42		73,5	73	0,5	0:02:08	Згурский В.В
	Тюхтерек	2019-05-16 4:50:49	2019-05-16 4:52:25		93,4	93	0,4	0:01:36	Тимаков Г.М.
	Тюхтерек	2019-05-16 4:56:04	2019-05-16 4:57:17		133	133	0	0:01:13	Сугреев В.А.
	Тюхтерек	2019-05-16 5:01:14	2019-05-16 5:03:08		218,3	218	0,3	0:01:54	Морозов Н.А
	Тюхтерек	2019-05-16 5:03:40	2019-05-16 5:06:05		228,2	228	0,2	0:02:25	Бардынов А.

Рисунок 44 – Накопительный отчет заправок техники АТЗ в Google Sheet

В данной версии отчета все бензовозы стабильно выгружаются в отчет, тем самым отдел информационной аналитики смог полностью перевести работу с данным отчетом в новую среду.

3.2.5 Описание бизнес–логики формирования отчета тяжелой и колесной технике

Логика работы данного отчета сложнее, чем предыдущего, так как кроме добавления данных построчно, появляется необходимость также выводить итоговые значения за отчетную неделю, в самих строках используется дополнительное форматирование и объединение ячеек, присутствует несколько отчетных таблиц, которые разбиваются по типу техники и закрепленному направлению.

Данный метод формирования отчета принимает данные из GetOnlineInfoAll и объект с информацией о ТС. В начале метода сразу идет ветвление по ключу CompanyID и Type объекта ТС. Относительно значения ID

направления определяется, с какой таблицей работать. Более того, при условии, что Type = WHEELVEH (Тип техники – Колесная) и ID направления соответствует направлению «Артель старателей Ангара–Север», проводится проверка на дополнительное условие AreaID – ID закрепленного участка. Если AreaID соответствует участку «Служебный Автотранспорт», то данная техника обрабатывается в таблице для Служебного транспорта. Такое ветвление обусловлено иерархией в схеме, исторически «Служебный автотранспорт» находился в направлении «Артель старателей Ангара–Север» (Рисунок 45), что является не корректным по отношению к реальным объектам.

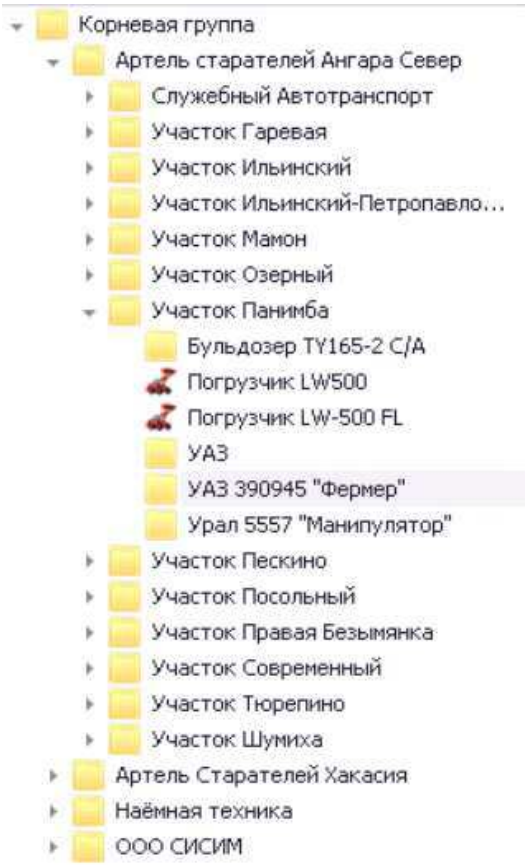


Рисунок 45 – Иерархия направлений и участков в схеме ООО «СпецПромАвтоматика»

Поиск листа и ключ ТС аналогичен отчету по заправкам техники АТЗ.

При отсутствии листа он создается и оформляется по шаблонным параметрам, ТС заносится в общий лист, и далее работа метода идет, как если бы лист был найден.

Если лист найден, идет запрос данных всего листа, и на основе количества строк определяется индекс последней строки. Стоит заметить, что в отчете АТЗ последняя строка забирается методом GoogleSheetsAPI.GetLastRow(), работа которого заключается в добавлении данных в первую незаполненную строку, получении индекса новой строки и последующем ее удалении. Тут используется метод GoogleSheetsAPI.GetAllSheetData(), который забирает весь лист и все данные, и

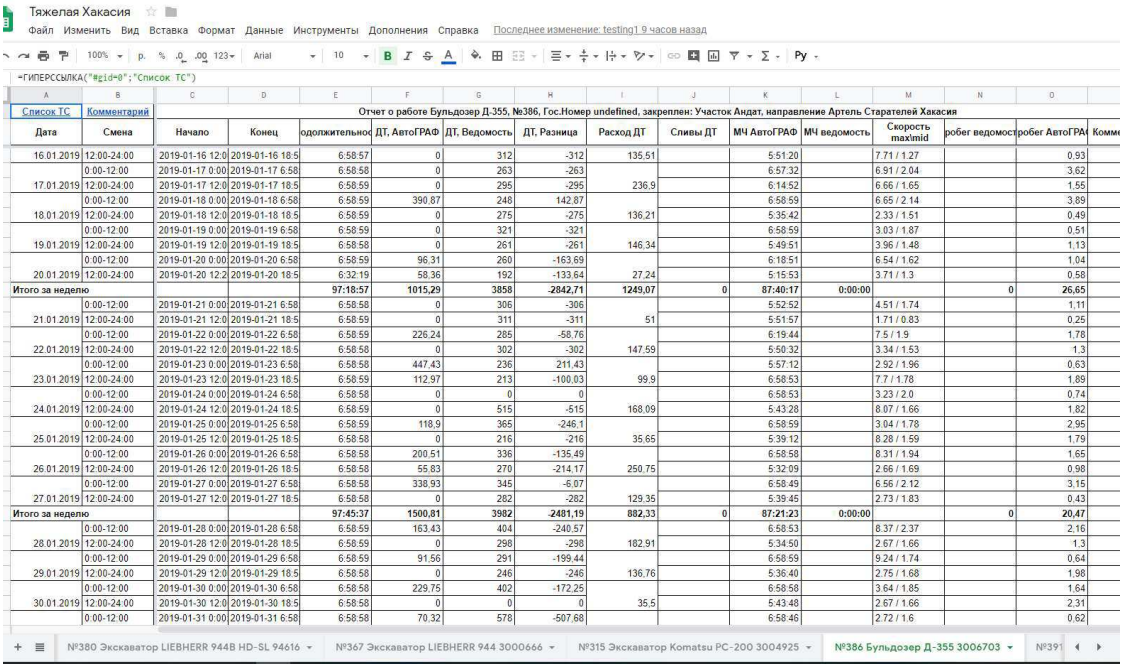
на основе длины данного массива строк получает номер последней строки. Данное различие связанно с ошибкой в самом GoogleSpreadSheets API. Без каких–либо предпосылок или закономерностей метод spreadsheets.values.append, который добавляет значения в электронную таблицу, и при этом значения будут добавлены после последней строки таблицы [24], иногда добавляет строки не после последней заполненной строки, а выше нее на одну или две строки. Соответственно, индекс новой строки определяется некорректно и метод GetLastRow() возвращает неправильный индекс последней строки. При этом, данная проблема была замечена только в данном виде отчета, в таблице с АТЗ все работает корректно. Запрос данных всего листа, с одной стороны, увеличивает нагрузку на сетевое соединение и увеличивает время данного метода (возможны большие объемы данных), с другой стороны, всегда однозначно определит размеры заполненной таблицы на листе.

Так как форма отчета предполагает итоговую строку в конце отчетного периода, а также сутки в данном отчете занимают две строки (первая и вторая смена), то необходимо получать данные последних трех строк. В данных строках снизу вверх ведется поиск даты последних заполненных суток. Также, на данном этапе происходит уточнение, является ли последняя строка итоговой. Эта проверка необходима для корректного заполнения формата текста. Дело в том, что метод, добавляющий новые данные, использует формат последней строки для всех новых строк. Например, если у последней строки был полужирный шрифт, все новые строки ниже тоже будут полужирными.

Дата с листа сравнивается с датой из GetOnlineInfoAll. При условии, что разница между датами более суток, происходит запрос значений, необходимых для данного отчета, пока разность дат синхронизации и последний полученных суток не станет меньше 24 часов. В отличии от первого отчета, в котором использовался запрос GetStage для получения необходимых данных, в данном отчете необходимо использовать такой метод AutoGRAPH.NET как GetTripTotal, он возвращает итоговые данные по ТС за выбранный период. При этом, для заполнения одних суток необходимо использовать два таких запроса, с временным интервалом 00:00:00–11:59:59 и 12:00:00–23:59:59. Когда запросы за одни сутки получены и перенесены в строки, в отдельную переменную заносятся шаблонный диапазон объединения ячеек со смещением последней строки, а также в другую переменную информация об обычной строке, если последняя строка в таблице – итоговая. В той же итерации идет проверка последней полученной даты на день недели. Если она соответствует воскресенью, то добавляется итоговая строка, охватывающая формулой суммирования неделю по каждому интересующему нас столбцу. Также, в отдельную переменную добавляется номер итоговой строки, чтобы после добавления данных на лист сделать шрифт данной строки полужирным. После выхода из цикла идет поочередная проверка переменных на наличие значений. В первую очередь проверяется переменная, в которой хранятся значения для таблицы. Если данные есть, они добавляются на лист, после задаются границы ячеек, объединяются ячейки, задается формат «Продолжительность» для столбцов с моточасами.

Далее проверяется переменная с координатами строк с обычным шрифтом. Если она не пустая (а это значит, что последняя строка была полужирной, и сейчас все новые строки в таблице имеют такой же шрифт), тогда происходит изменение шрифта на обычный для всех добавленных строк. В конце, проверяется переменная, в которой должны храниться координаты итоговой строки. Если переменная не пустая, тогда циклом отправляются запросы изменить формат на полужирный шрифт с координатами итоговой строки. Цикл будет идти по всем объектам в переменной.

При данной версии отчета таблица выглядит как на рисунке 46. Здесь инженер–аналитик уже занес данные из ведомостей.



Тяжелая Хакасия

Файл Изменить Вид Вставка Формат Данные Инструменты Дополнения Справка Последнее изменение: testing1 9 часов назад

=ГИПЕРССЫЛКА("#gid=0";"Список ТС")

Список ТС	Комментарий	Отчет о работе Бульдозер Д-355, №386, Гос.Номер undefined, закреплен: Участок Андат, направление Артель Старателей Хакасия													
Дата	Смена	Начало	Конец	одолжительнос	ДТ, АвтоГРАФ	ДТ, Ведомость	ДТ, Разница	Расход ДТ	Сливы ДТ	МЧ АвтоГРАФ	МЧ ведомость	Скорость max\mid	робег ведомост	робег АвтоГРА	Комме
16.01.2019	12:00-24:00	2019-01-16 12:0	2019-01-16 18:5	6:58:57	0	312	-312	135,51		5:51:20		7.71 / 1.27		0,93	
	0:00-12:00	2019-01-17 0:00	2019-01-17 6:58	6:58:58	0	263	-263			6:57:32		6.91 / 2.04		3,62	
17.01.2019	12:00-24:00	2019-01-17 12:0	2019-01-17 18:5	6:58:59	0	295	-295	236,9		6:14:52		6.66 / 1.65		1,55	
	0:00-12:00	2019-01-18 0:00	2019-01-18 6:58	6:58:59	390,87	248	142,87			6:58:59		6.65 / 2.14		3,89	
18.01.2019	12:00-24:00	2019-01-18 12:0	2019-01-18 18:5	6:58:59	0	275	-275	136,21		5:35:42		2.33 / 1.51		0,49	
	0:00-12:00	2019-01-19 0:00	2019-01-19 6:58	6:58:59	0	321	-321			6:58:59		3.03 / 1.87		0,51	
19.01.2019	12:00-24:00	2019-01-19 12:0	2019-01-19 18:5	6:58:58	0	261	-261	146,34		5:49:51		3.96 / 1.48		1,13	
	0:00-12:00	2019-01-20 0:00	2019-01-20 6:58	6:58:59	96,31	260	-163,69			6:18:51		6.54 / 1.62		1,04	
20.01.2019	12:00-24:00	2019-01-20 12:2	2019-01-20 18:5	6:32:19	58,36	192	-133,64	27,24		5:15:53		3.71 / 1.3		0,58	
Итого за неделю				97:18:57	1015,29	3858	-2842,71	1249,07	0	87:40:17	0:00:00		0	26,65	
	0:00-12:00	2019-01-21 0:00	2019-01-21 6:58	6:58:58	0	306	-306			5:52:52		4.51 / 1.74		1,11	
21.01.2019	12:00-24:00	2019-01-21 12:0	2019-01-21 18:5	6:58:59	0	311	-311	51		5:51:57		1.71 / 0.83		0,25	
	0:00-12:00	2019-01-22 0:00	2019-01-22 6:58	6:58:59	226,24	285	-58,76			6:19:44		7.5 / 1.9		1,78	
22.01.2019	12:00-24:00	2019-01-22 12:0	2019-01-22 18:5	6:58:58	0	302	-302	147,59		5:50:32		3.34 / 1.53		1,3	
	0:00-12:00	2019-01-23 0:00	2019-01-23 6:58	6:58:58	447,43	236	211,43			5:57:12		2.92 / 1.96		0,63	
23.01.2019	12:00-24:00	2019-01-23 12:0	2019-01-23 18:5	6:58:59	112,97	213	-100,03	99,9		6:58:53		7.7 / 1.78		1,89	
	0:00-12:00	2019-01-24 0:00	2019-01-24 6:58	6:58:58	0	0	0			6:58:53		3.23 / 2.0		0,74	
24.01.2019	12:00-24:00	2019-01-24 12:0	2019-01-24 18:5	6:58:59	0	515	-515	168,09		5:43:28		8.07 / 1.66		1,82	
	0:00-12:00	2019-01-25 0:00	2019-01-25 6:58	6:58:59	118,9	365	-246,1			6:58:59		3.04 / 1.78		2,95	
25.01.2019	12:00-24:00	2019-01-25 12:0	2019-01-25 18:5	6:58:58	0	216	-216	35,65		5:39:12		8.28 / 1.59		1,79	
	0:00-12:00	2019-01-26 0:00	2019-01-26 6:58	6:58:58	200,51	336	-135,49			6:58:58		8.31 / 1.94		1,65	
26.01.2019	12:00-24:00	2019-01-26 12:0	2019-01-26 18:5	6:58:58	55,83	270	-214,17	250,75		5:32:09		2.66 / 1.69		0,98	
	0:00-12:00	2019-01-27 0:00	2019-01-27 6:58	6:58:58	338,93	345	-6,07			6:58:49		6.56 / 2.12		3,15	
27.01.2019	12:00-24:00	2019-01-27 12:0	2019-01-27 18:5	6:58:58	0	282	-282	129,35		5:39:45		2.73 / 1.83		0,43	
Итого за неделю				97:45:37	1500,81	3982	-2481,19	882,33	0	87:21:23	0:00:00		0	20,47	
	0:00-12:00	2019-01-28 0:00	2019-01-28 6:58	6:58:59	163,43	404	-240,57			6:58:53		8.37 / 2.37		2,16	
28.01.2019	12:00-24:00	2019-01-28 12:0	2019-01-28 18:5	6:58:59	0	298	-298	182,91		5:34:50		2.67 / 1.66		1,3	
	0:00-12:00	2019-01-29 0:00	2019-01-29 6:58	6:58:59	91,56	291	-199,44			6:58:59		9.24 / 1.74		0,64	
29.01.2019	12:00-24:00	2019-01-29 12:0	2019-01-29 18:5	6:58:58	0	246	-246	136,76		5:36:40		2.75 / 1.68		1,98	
	0:00-12:00	2019-01-30 0:00	2019-01-30 6:58	6:58:58	229,75	402	-172,25			6:58:58		3.64 / 1.85		1,64	
30.01.2019	12:00-24:00	2019-01-30 12:0	2019-01-30 18:5	6:58:58	0	0	0	35,5		5:43:48		2.67 / 1.66		2,31	
	0:00-12:00	2019-01-31 0:00	2019-01-31 6:58	6:58:58	70,32	578	-507,68			6:58:46		2.72 / 1.6		0,62	

№380 Экскаватор LIEBHERR 944B HD-SL 94616 | №367 Экскаватор LIEBHERR 944 3000666 | №315 Экскаватор Komatsu PC-200 3004925 | №386 Бульдозер Д-355 3006703 | №391

Рисунок 46 – Новый отчёт по колёсной и тяжёлой технике Холдинга

Данный отчет так же был предоставлен отделу информационной аналитики, и после тестирования и исправления мелких недочетов в нем началась полноценная работа.

3.2.5 Описание бизнес–логики формирования отчета по подрядчикам

Отчет по технике подрядчиков использует более сложную логику, чем остальные отчеты, поэтому он выполняется в последнюю очередь. Он является самым трудоемким из всех реализованных отчетов по следующим причинам:

– лист в таблице – это вся техника подрядчика, на листе располагается несколько ТС. Соответственно определение диапазона получения и внесения данных, а также какая именно единица расположена на данном участке таблицы, становится более сложной в реализации;

– отчетный период у данного отчета – месяц, каждый новый месяц необходимо заполнять новую таблицу;

- необходимо предоставлять данные в отчет актуального месяца, так и прошлого. Существует вероятность дозагрузки данных с прошлого месяца;
- подрядчики не имеют отдельного типа в схеме Автограф. Список ТС определяется данными в аналитической записке по подрядчикам холдинга «СибЗолото»;
- у разных подрядчиков время смен различается;
- необходимо учитывать статус ДУТ из аналитической записки по подрядчикам;
- в данном отчете необходимо создавать таблицы под ТС, которые не оборудованы ССМ;
- для создания одной строки по ТС необходимо запросить разные типы запросов;
- идентификация ТС происходит по номеру из 1С, указанных в аналитической записке по Подрядчикам;
- различия количества дней в разным месяцах;
- данные из сводной таблицы необходимо забирать по листу опрашиваемого месяца (если идет работа с отчетом за март, а сегодня 5 апреля, то данные в аналитической записке по подрядчикам необходимо брать на вкладке «Март 2», а не «Март 1» или «Апрель 1»).

Данные требования необходимо было учитывать при реализации отчета, соответственно на него было потрачено большая часть времени разработки.

При использовании данного способа данные у каждого ТС из метода GetOnlineInfoAll, а также список объектов мониторинга, как и отчеты, описанные ранее. У данного метода реализованы следующие внутренние функции:

- SearchWorksheet() - метод ищет необходимую таблицу для обработки, возвращает id искомой таблицы;
- StartUpdating() – принимает данные из GetOnlineInfoAll, список объектов мониторинга и рабочую дату. Данная функция отвечает за добавление данных в отчет. У данной функции также есть свои, внутренние функции:
- CreateNewTableOfConstactorCar() – функция, которая создает и форматирует область таблицы под ТС;
- AppendNewDataInTableConstractorTC() – метод проверяет наличие и добавляет новые данные для выбранного подрядчика. У данной функции также есть функция:
- AppendDataInWorkingSheet() – функция добавляет данные в таблицу с ТС подрядчика.

При начале работы метода объявляется переменная how_many_actual_month_used, которая указывает, сколько месяцев опрашивать. Алгоритм обрабатывает два месяца – тот, который идет сейчас и прошлый месяц. После запрашиваем сервисную таблицу, в которой указаны ID созданных таблиц, год, и месяц для этой таблицы. Далее получаем актуальный объект datetime (дату и время). Далее идет условие how_many_actual_month_used != 0, при невыполнении данного условия метод завершает свою работу. В начале работы метода данное условие выполняется.

Внутри этого блока if имеется цикл, с количеством итераций, равной переменной how_many_actual_month_used. Счетчик объявлен как difference_in_month. Далее идет выражение:

working_datetime = actual_datetime - datetime.timedelta(weeks= difference_in_month*4)

if difference_in_month != 0:

working_datetime = datetime.datetime(working_datetime.year, working_datetime.month, 25)

Данный код необходим для установления рабочей даты для отчетов за прошлые месяцы. После определения рабочей даты вызывается функция StartUpdating().

В начале функции StartUpdating() из рабочей даты формируется название нужного нам листа в аналитической записке по Подрядчикам. После этого, с данного листа забираем 19 столбцов, начиная от названия подрядчика, заканчивая временем смены. Так как данные приходят из таблицы списком списков, целесообразно уточнить важные для создания отчета индексы вложенных списков (строк таблиц):

- индекс 0 – имя подрядчика (если значения нет, значит имя подрядчика выше по строкам);
- индекс 2 – участок (если значения нет, значит участок выше по строкам);
- индекс 4 – модуль Автограф (данное значение необходимо дополнительно проверять на валидность);
- индекс 5 – бортовой номер (это название машины, по данному значению будет производиться поиск на рабочем листе);
- индекс 6 – ДУТ (необходим для корректировки значений при заполнении данными топлива):
- * - ДУТ не работает;
- ** - ДУТ висит после заправки, но возвращается ч\з не продолжительное время;
- *** - требуется тарировка ДУТа;
- **** - проблема с антенной, наблюдаются частые периоды отсутствия сигнала;
- ** *** - с периодичностью пропадает сигнал в движении;
- н/р - не было рейсов;
- н/д - нет данных.
- Индекс 17 – включать ли ТС в отчет (да/нет);
- Индекс 18 – время пересменки (относительно этой переменной корректируется начало и конец смены ТС).

Поле получения этих данных, инициализируем и присваиваем None переменной constractor_name – в данной переменной будет храниться название подрядчика – название рабочего листа.

Также инициализирует и присваиваем None переменной work_sheet – в ней будет расположен объект с данными листа, в котором будет работать код.

После получения информации обо всех листах в отчете по подрядчикам, инициализируем переменную days_in_work_month и заносим туда значения количества дней в рабочем месяце.

Далее циклом проходим по каждой строке из аналитической записки. В начале есть важная конструкция try\except, при возникновении ошибки в котором производится выход из функции StartUpdate(), так как список машин подрядчиков кончился.

Если ошибки не произошло, инициализируем переменную autograph_empty и присваиваем ей значение False – данная переменная будет определять поведение заполнения ТС, если единицы не будет номера Автографа, а включить машину в отчет необходимо.

Далее идет конструкция if, условием которой является наличие названия подрядчика в строке. Если название подрядчика есть в данной строке, тогда в этой конструкции присваивается переменной constractor_name имя нового подрядчика, а переменной work_sheet присваивается None. Далее идет цикл for, в котором происходит поиск листа в рабочей таблице нового подрядчика. Если лист найден, тогда информация о нем присваивается work_sheet. После цикла идет конструкция try\except, которая позволяет перейти к следующему ТС, если данное ТС не должно быть включена в отчет (по индексу 17). После данной конструкции идет проверка переменной work_sheet на отсутствие данных (None). Если условие истинно, то в этой конструкции происходит создание листа для данного подрядчика, который указан в переменной constractor_name, и в той же конструкции добавляется строка с информацией о новом листе подрядчика на общий лист в рабочей таблице и обнуляем переменную row_tc_in_sheet, в которой должен располагаться список с бортовыми номерами ТС на рабочем листе.

После кода выше у нас в любом случае имеется рабочий лист с подрядчиком. Далее мы обнуляем переменную tc_id, в котором будем хранить ID машины, которая сейчас находится в итерации цикла for. Далее в конструкции try пытаемся получить валидное значение номера модуля Автограф. Если возникает ошибка, это означает, что значение некорректное. В таком случае переменной autograph_empty, которая определяет поведение создания таблицы ТС при отсутствии модуля Автограф, присваиваем значение True. Далее идет конструкция if с условием отсутствия значения на позиции, где должен быть номер модуля Автограф. Если условие выполняется, то в теле if также переменной autograph_empty присваиваем значение True. После этого снова идет конструкция if, с условием autograph_empty = False. Если условие выполняется, значит у данного ТС есть номер Автограф. В этой конструкции if происходит поиск ID данного объекта по номеру модуля Автограф, обернутый в try\except. Если ID найдено, оно присваивается в переменную tc_id. Если возникает ошибка типа ValueError, тогда autograph_empty присваиваем значение True.

Кодом выше мы однозначно определили наличие или отсутствие данной единицы в схеме. Следующей строкой идет проверка состояния переменной autograph_empty. Если значение False, тогда в теле if ищем дату последних

данных в GetOnlineInfoAll, при этом поиск также обернут в блок try. Если поиск прошел успешно, тогда в переменной get_online_info_date находится объект datetime с информацией о последних данных по данному ТС. Если в ходе поиска возникла ошибка, тогда переменной autograph_empty присваиваем значение True.

После этого идет конструкция try\except\else, в блоке try у которого только if с условием row_tc_in_sheet == None. Если условие истинно и переменной, в которой хранятся номера ТС на рабочем листе ничего нет, тогда пытаемся присвоить row_tc_in_sheet значения из рабочего листа с номерами ТС. Если возникает ошибка KeyError. Это означает, что на листе с подрядчиком еще нет ни одной таблицы с ТС. В случае возникновения данного исключения, переменной rose_of_wind присваиваем число 2. rose_of_wind – переменная, которая указывает левый верхний угол таблицы рабочего ТС. Так как на рабочем листе с подрядчиком ТС еще нет, тогда rose_of_wind устанавливается в начало (значение 2 – это 2 столбец, в первом столбце по формату данного отчета располагаются даты рабочего месяца). В этом же блоке except, после присвоения значения rose_of_wind, вызывается метод CreateNewTableOfConstactorCar(), который в качестве аргументов принимает:

- rose_of_wind – позицию ТС на листе подрядчика;
- ТС – строка из аналитической записки подрядчиков, в которой содержатся все данные по технике подрядчика;
- work_spreadsheet – ID рабочей таблицы;
- work_sheet – объект с данными по актуальному рабочему листу;
- working_datetime – рабочая дата;

CreateNewTableOfConstactorCar() – данная функция создает на рабочем листе подрядчика новую таблицу с рабочим ТС.

В начале функции переменной header_text присваивается результат работы метода Reports.GenerateHeaderLines(), с аргументами переменной ТС и значением constractor = True. Данный метод возвращает стандартный заголовок таблицы, аргументами данного метода нужно указывать, для какого именно отчета необходим заголовок таблицы. Далее происходит вставка этой шапки в позицию rose_of_wind по столбцу, и во вторую строку. После этого происходят шаблонные объединения ячеек, также со смещением на переменную rose_of_wind, форматирование заголовкам таблицы методом GoogleSheetsAPI.SetHeaderFormat().

После первичного форматирования таблицы происходит создания списка дат к строкам слева от созданной таблицы, а также слова «Итого» под последним днем месяца. После этого происходит заполнение формулами строки «Итого». Все эти данные записываются в переменную, которую сразу после заполнения последней строки добавляют на рабочий лист. В последней части данной функции происходит конечное форматирование созданной таблицы (границы, полужирный шрифт). После этого происходит возврат на место, где данная функция была вызвана.

Если ошибки KeyError не возникло в последнем блоке try, тогда переходим в блок else (блок else в конструкции try\except\else выполняется

тогда, когда блок try завершился без ошибок). В данном блок находится конструкция try/except. В блоке try присваивается переменной rose_of_wind значение индекса по бортовому номеру итерируемой ТС и прибавляется еще единица. Если при данной операции возникает ошибка ValueError (ТС на листе подрядчика есть, но рабочего ТС нету), тогда происходит повторное присвоение значений переменной row_tc_in_sheet, переменной rose_of_wind присваивается значение длины списка row_tc_in_sheet плюс 10. После этих операций вызывается метод CreateNewTableOfConstactorCar() с теми же входными параметрами и поведением, которые указаны выше.

В последней части кода функции StartUpdate(), в которую можно попасть, если в последнем блоке try ошибки не произошло (в переменную rose_of_wind корректно присвоилось значение бортового номера рабочего ТС плюс один) имеется блок if с условием autograph_empty == False. Если условие истинно (это значит, что итерируемое ТС есть в схеме, имеет таблицу на рабочем листе подрядчика, есть дата последних данных и его нужно включать в отчет), тогда вызывается функция AppendNewDataInTableConstractorTC(), которое принимает следующие аргументы:

- rose_of_wind – позиция ТС в рабочей таблице подрядчика;
- ТС - строка из аналитической записки подрядчиков, в которой содержатся все данные по технике подрядчика;
- tc_id – ID итерируемого ТС;
- work_spreadsheet – рабочая таблица;
- work_sheet – Рабочий лист;
- get_online_info_date – данные из GetOnlineInfoAll;
- working_datetime – рабочая дата.

AppendNewDataInTableConstractorTC() – Метод проверяет наличие новых данных по ТС и при необходимости добавляет их.

В данном методе первая конструкция кода - try\except. В блоке try происходит присваивание переменной days_in_working_month количество дней в рабочем месяце, переменной last_day_sync дату последних данных в таблице итерируемого ТС. Если произошла ошибка KeyError, значит, что данных по ТС в этом месяце еще не было в рабочей таблице, соответственно переменой last_day_sync присваивается значение 0.

После этого блока расположена конструкция if с условием days_in_working_month = last_day_sync, что означает, что все данные по рабочему ТС заполнены на рабочем листе. Если условие истинно, происходит возврат функции с помощью return, и цикл переходит к следующему рабочему–итерируемому ТС.

Если условие ложно для блока выше, тогда разбивается время смены на часы и минуты в отдельные переменные, происходит присваивание переменной SD значения datetime рабочей даты, но со временем, относительно времени смены). Сразу ниже аналогичное присвоение переменной SD_GETSTAGE значения рабочей даты, только вместо времени смены указаны нули. После получения переменных с корректными начальными значениями времени происходит сравнение на полноту данных в сутках в блоке while. Если условие

ложно (истинность условия предполагает наличие данных за смену минимум за 1 сутки), тогда блок while пропускается. В данной функции после цикла whilr остается только условие проверки ДУТ на значение «***» (требуется тарировка). Если данное условие выполняется, тогда под таблицей ТС в обговоренном месте указывается комментарий «Возможно, требуется тарировка ДУТа». Также и после цикла while есть возможность попасть в последний блок if.

В ином случае, если значения условия истинно, и данные как минимум за 2 смены есть, тогда в теле цикла while имеется конструкция if с условием SD.month != working_datetime.month. Если условие истинно, собранные данные будут занесены на лист, функция вернет return и цикл будет начнется для новой единицы техники подрядчика. Данная конструкция расположена здесь для страховки от переполнения данных в таблице, когда опрос идет с не актуального месяца.

Если данный блок пройден, далее инициируются переменные:

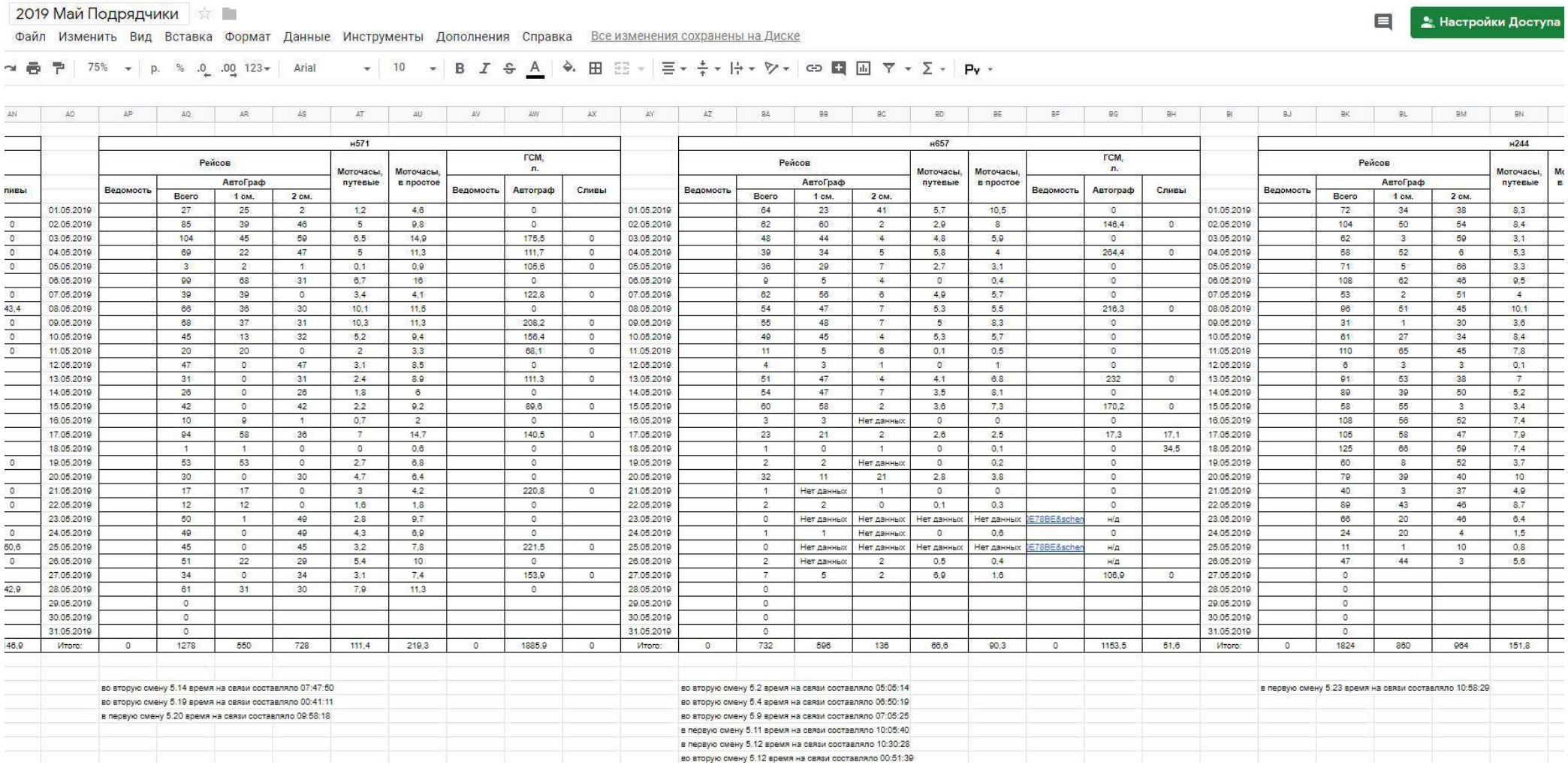
- ED_GETSTAGE – как SD_GETSTAGE плюс одни сутки;
- SD_first – как и SD;
- ED_first - как SD_first плюс 11 часов, 59 минут, 59 секунд;
- SD_second – как ED_first + 1 секунда;
- ED_second – как SD_second плюс 11 часов, 59 минут, 59 секунд;

Данные переменные нужны для корректных временных запросов к AutoGRAPH.NET.

После инициализации данных переменных запрашиваются необходимые запросы (GetStage за сутки, GetTripTotal за сутки, и два GetTripTotal за первую и вторую смены). После получения всех запросов в двух конструкциях try\except последовательно проверяется время на связи модуля за первую и вторую смены. Если время на связи менее 11 часов, то, по договоренности с отделом информационной аналитики, ниже таблицы с ТС подрядчика указывается комментарий, сколько времени в данную смену работала единицы. При возникновении ошибок в получении этого значения указываем, что в данную смену данных не было.

Далее из запросов собирается строка с данными из запросов по моточасам. После этого из ответа метода GetStage получаем информацию о сливах и заправках ТС за сутки. По индексу 7, описанному выше вместо значений может быть комментарий «н\д по ППО». После обработки данных по топливу, также конструкцией if статуса ДУТ и наличия там значения «***» в определенном месте ниже таблицы с ТС указывается комментарий «Возможно, требуется таррировка ДУТ». После этого вызывается метод Reports.AppendDataInWorkingSheet(). Данный метод заносит все обработанные данные в таблицу с ТС. Если все данные получены до последних из GetOnlineInfoAll, или рабочий месяц закончился, обрабатывается следующий подрядчик по тому же алгоритму, описанному выше. В момент обработки всех подрядчиков в рабочем месяце, метод переходит к следующему месяцу. Когда обработаны все необходимые месяцы, метод завершает свою работу.

Отчет по подрядчикам показан на рисунке 47.



		н571										н657										н244						
		Рейсов				Моточасы, путевые	Моточасы, в простое	ГСМ, л.				Рейсов				Моточасы, путевые	Моточасы, в простое	ГСМ, л.				Рейсов				Моточасы, путевые	
ливы		Ведомость	АвтоГраф					Ведомость	Автограф	Сливы		Ведомость	АвтоГраф					Ведомость	Автограф	Сливы		Ведомость	АвтоГраф				
			Всего	1 см.	2 см.								Всего	1 см.	2 см.								Всего	1 см.	2 см.		
	01.05.2019		27	25	2	1,2	4,6		0		01.05.2019		64	23	41	5,7	10,5		0		01.05.2019		72	34	38	8,3	
0	02.05.2019		85	39	46	5	9,8		0		02.05.2019		62	60	2	2,9	8		146,4	0	02.05.2019		104	50	54	8,4	
0	03.05.2019		104	45	59	6,5	14,9		175,5	0	03.05.2019		48	44	4	4,8	5,9		0		03.05.2019		62	3	59	3,1	
0	04.05.2019		69	22	47	5	11,3		111,7	0	04.05.2019		39	34	5	5,8	4		264,4	0	04.05.2019		58	52	6	5,3	
0	05.05.2019		3	2	1	0,1	0,9		105,6	0	05.05.2019		36	29	7	2,7	3,1		0		05.05.2019		71	5	66	3,3	
	06.05.2019		99	68	31	6,7	16		0		06.05.2019		9	5	4	0	0,4		0		06.05.2019		108	62	46	9,5	
0	07.05.2019		39	39	0	3,4	4,1		122,8	0	07.05.2019		62	56	6	4,9	5,7		0		07.05.2019		53	2	51	4	
43,4	08.05.2019		66	36	30	10,1	11,5		0		08.05.2019		54	47	7	5,3	5,5		216,3	0	08.05.2019		96	51	45	10,1	
0	09.05.2019		68	37	31	10,3	11,3		208,2	0	09.05.2019		55	48	7	5	8,3		0		09.05.2019		31	1	30	3,6	
0	10.05.2019		45	13	32	5,2	9,4		156,4	0	10.05.2019		49	45	4	5,3	5,7		0		10.05.2019		61	27	34	8,4	
0	11.05.2019		20	20	0	2	3,3		68,1	0	11.05.2019		11	5	6	0,1	0,5		0		11.05.2019		110	65	45	7,8	
	12.05.2019		47	0	47	3,1	8,5		0		12.05.2019		4	3	1	0	1		0		12.05.2019		6	3	3	0,1	
	13.05.2019		31	0	31	2,4	8,9		111,3	0	13.05.2019		51	47	4	4,1	6,8		232	0	13.05.2019		91	53	38	7	
	14.05.2019		26	0	26	1,8	6		0		14.05.2019		54	47	7	3,5	8,1		0		14.05.2019		89	39	50	5,2	
	15.05.2019		42	0	42	2,2	9,2		89,6	0	15.05.2019		60	58	2	3,6	7,3		170,2	0	15.05.2019		58	55	3	3,4	
	16.05.2019		10	9	1	0,7	2		0		16.05.2019		3	3	Нет данных	0	0		0		16.05.2019		108	56	52	7,4	
	17.05.2019		94	58	36	7	14,7		140,5	0	17.05.2019		23	21	2	2,6	2,5		17,3	17,1	17.05.2019		105	58	47	7,9	
	18.05.2019		1	1	0	0	0,6		0		18.05.2019		1	0	1	0	0,1		0	34,5	18.05.2019		125	66	59	7,4	
0	19.05.2019		53	53	0	2,7	6,8		0		19.05.2019		2	2	Нет данных	0	0,2		0		19.05.2019		60	8	52	3,7	
	20.05.2019		30	0	30	4,7	6,4		0		20.05.2019		32	11	21	2,8	3,8		0		20.05.2019		79	39	40	10	
0	21.05.2019		17	17	0	3	4,2		220,8	0	21.05.2019		1	Нет данных	1	0	0		0		21.05.2019		40	3	37	4,9	
0	22.05.2019		12	12	0	1,6	1,8		0		22.05.2019		2	2	0	0,1	0,3		0		22.05.2019		89	43	46	8,7	
	23.05.2019		50	1	49	2,8	9,7		0		23.05.2019		0	Нет данных	Нет данных	Нет данных	Нет данных	IE78BE&schen	н/д		23.05.2019		66	20	46	6,4	
0	24.05.2019		49	0	49	4,3	6,9		0		24.05.2019		1	1	Нет данных	0	0,6		0		24.05.2019		24	20	4	1,5	
60,6	25.05.2019		45	0	45	3,2	7,8		221,5	0	25.05.2019		0	Нет данных	Нет данных	Нет данных	Нет данных	IE78BE&schen	н/д		25.05.2019		11	1	10	0,8	
0	26.05.2019		51	22	29	5,4	10		0		26.05.2019		2	Нет данных	2	0,5	0,4		н/д		26.05.2019		47	44	3	5,6	
	27.05.2019		34	0	34	3,1	7,4		153,9	0	27.05.2019		7	5	2	6,9	1,6		106,9	0	27.05.2019		0				
42,9	28.05.2019		61	31	30	7,9	11,3		0		28.05.2019		0								28.05.2019		0				
	29.05.2019		0								29.05.2019		0								29.05.2019		0				
	30.05.2019		0								30.05.2019		0								30.05.2019		0				
	31.05.2019		0								31.05.2019		0								31.05.2019		0				
46,9	Итого:	0	1278	550	728	111,4	219,3	0	1885,9	0	Итого:	0	732	596	136	66,6	90,3	0	1153,5	51,6	Итого:	0	1824	860	964	151,8	

во вторую смену 5.14 время на связи составляло 07:47:50
во вторую смену 5.19 время на связи составляло 00:41:11
в первую смену 5.20 время на связи составляло 09:58:18

во вторую смену 5.2 время на связи составляло 05:05:14
во вторую смену 5.4 время на связи составляло 06:50:19
во вторую смену 5.9 время на связи составляло 07:05:25
в первую смену 5.11 время на связи составляло 10:05:40
в первую смену 5.12 время на связи составляло 10:30:28
во вторую смену 5.12 время на связи составляло 00:51:39

в первую смену 5.23 время на связи составляло 10:58:29

Рисунок 47 – Сформированный отчет по подрядчикам

После корректировки параметров в самой системе «АвтоГРАФ», с новым отчетом стал работать инженер–аналитик.

3.3 Расчет экономической эффективности внедренных изменений

Экономическая выгода проявляется из–за уменьшения времени на формирование отчетности. В таблице 19 представлена сводная информация о затратах на каждый отчет из второй главы.

Таблица 19 – Сводная информация о выгоде после внедрения ИАС

Показатели	Отчет		
	Отчет по подрядчикам	Отчет по заправкам техники АТЗ	Отчет по тяжелой и колесной технике
Временные затраты до изменений на отчет, часы в месяц	103	72,5	89
Стоимостные затраты до внедрения ИАС, руб.	25750	20234,375[1]	22250
Временные затраты после внедрения ИАС, часы	91,67	55	27,59
На сколько уменьшилось время формирования отчетности после внедрения ИАС, %	11	24	69
Стоимостные затраты после внедрения ИАС, руб.	22918	15860[2]	6898
Выгода от внедрения, руб.	2833	4375	15353
1 – рассчитывалось на основе 50 часов времени аналитика и 22,5 часов времени старшего аналитика. 2 – рассчитывалось на основе 32,5 часов времени аналитика и 22,5 часов времени старшего аналитика.			

Итого в денежном эквиваленте компания может экономить 22561 рубль в месяц на формирование этой отчетности при устоявшемся парке оборудованной техники.

В соответствии с целями из пункта 2.3 необходимо произвести расчет количества единиц техники, которые можно оборудовать модулями ССМ и их смогут обрабатывать сотрудники отдела информационной аналитики. В таблице 20 представлена информация возможности увеличения обрабатываемых ТС при прочих равных условиях. Расчет времени на один объект мониторинга производился из описанных действий на каждую единицу в определенном отчете из пунктов 2.4.1–2.4.3.

Таблица 20 – расчет возможности добавления новых объектов мониторинга.

Показатели	Отчет по подрядчикам	Отчет по заправкам техники АТЗ	Отчет по тяжелой и колесной технике
Количество объектов мониторинга в отчете	70	32	140
Время на одну единицу техники до внедрения ИАС, сек.	5275	8313	2257
Время на одну единицу техники после внедрения ИАС, сек.	4699	6362	665
Время на отчет до внедрения ИАС, сек	370510	261150	320880
Время на отчет после внедрения ИАС, сек.	328950	198690	98000
Освободившееся время сотрудников, сек.	41560	62460	222880
Количество новой техники, которое потребуется для возвращения в первоначальные условия	9	10	335

Услуги информационной аналитики подрядчики оплачивают напрямую, и это является доходом компании. Можно приблизительно посчитать, что при стоимости в среднем 800 рублей в месяц за 1 ТС отчет по подрядчикам приносит 56000 рублей в месяц. После внедренных изменений возможно обрабатывать дополнительно 9 ТС тем же сотрудником, что может принести дополнительно 7200 рублей, при условии новых единиц тподключенной техники. Также, оборудование 335 единиц колесной и тяжелой техники холдинга в ближайшей перспективе маловероятно, соответственно освободившееся время другого специалиста можно будет использовать в отчете по подрядчикам. При условии активного подключения подрядчиков возможно увеличить доход на 37600 рублей данным отчетом, или новыми 47 ТС, которыми может заниматься аналитик, ответственный за отчет по колесной и тяжелой технике холдинга. Расчет времени в ущерб подключения новых бензовозов нецелесообразно, так как холдинг активно открывает новые участки золотодобычи и закупает новые бензовозы.

Расчет затрат на разработку информационно–аналитической системы рассчитывается по формуле:

$$З_{РИАС} = З_{ФОТР} + З_{ЭВМ} + З_{ССП} + Р_Н \quad (5)$$

Где $З_{ФОТР}$ – общий фонд оплаты труда разработчиков;

$З_{ЭВМ}$ – затраты, связанные с эксплуатацией техники;

$З_{ССП}$ – затраты на специальные программные продукты, необходимые для разработки ИАС;

$Р_Н$ – накладные расходы (обычно 30% от $З_{ФОТР}$).

Расчет фонда оплаты труда:

$$З_{ФОТР} = \sum_{j=1}^{m} О_{Р_j} * Т_{РПР_j} \quad (6)$$

Где $О_{Р_j}$ – Оклад j–го разработчика;

$Т_{РПР_j}$ – Общее время работы над ИАС.

Разработкой информационно–аналитической системы занимался один сотрудник отдела разработки и сопровождения ПО, с окладом 27000 рублей в месяц. С учетом НДФЛ 13%, «пенсионными» взносами 22%, медстрахованием 5,1%, соцстрахованием 2,9%, страхованием от несчастных случаев 0,2% в месяц предприятие тратит 38664 рубля, час времени сотрудника РСПО стоит 241,65 рубля.

На разработку было затрачено 120 часов рабочего времени, соответственно, фонд оплаты труда составляет 28998 рублей.

Затраты на эксплуатацию техники при разработке отсутствуют. Затрат на специальные программные продукты не было, все технологии в ИАС бесплатные. Накладные расходы составляют 8699,4 рубля.

Итого на разработку ИАС было потрачено 37697,4 рубля. Затрат на внедрение ИАС для компании не было.

Ожидаемая условно–годовая экономия от внедрения системы рассчитывается по формуле:

$$Э_г = С_1 - С_2 \ (7)$$

Где $Э_г$ – годовая экономия, рублей;

$С_1$– годовые текущие затраты до внедрения системы, рублей;

$С_2$ –годовые текущие затраты после внедрения системы, рублей;

На функцию F1 отдела информационной аналитики до внедрения ИАС использовалось 103812,5 рублей в месяц, или 1245750 рублей в год. После внедрения изменений затраты составляют 78620,63 рубля в месяц, или 943447,5 рублей в год. Соответственно, условно–годовая экономия составляет 302302,5 рубля в год.

ИАС приблизило достижение задачи из параграфа 2.3 – увеличить эффективность отдела информационной аналитики на 25%, так как функции формирования различной отчетности и получения раздаточных ведомостей занимало у отдела ИА 539 часов в месяц, а после изменений 439 часа, соответственно эффективность увеличилась на 18,55%.

Окупаемость системы рассчитывается следующим образом:

$$Т_{ок} = \frac{З_{РИАС}}{Э_г} \ (8)$$

Таким образом, срок окупаемости:

$$\frac{37697.4 \text{ рубля}}{302302{,}5 \text{ рублей в год}} = 0{,}1247 \text{ года} = 45{,}5 \text{ дня}$$

Из проведенных расчетов можно сказать, что разработанная ИАС экономически выгодна, поскольку срок окупаемости составляет всего полтора месяца, а экономический эффект составляет более 302 тысяч рублей в год.

В заключении можно сказать, что дальнейшая разработка информационно–аналитической системы на предприятии в плане добавления новых автоматизированных отчетов позволит и дальше увеличивать эффективность отдела аналитики, а также снижать себестоимость сформированной отчетности.

ЗАКЛЮЧЕНИЕ

Одним из основных условий успешной работы предприятия является автоматизация производства и снижение себестоимости ценностей, которые предприятие генерирует.

В результате выполнения дипломной работы были достигнуты следующие цели:

- изучена деятельность предприятия;
- рассмотрена организационная структура фирмы;
- изучен комплекс программных продуктов, использующихся на предприятии;
- описаны стратегические цели предприятия и задачи, способствующие их выполнению;
- проведен анализ проблемных зон, мешающих достижению целей компании и описаны их бизнес–процессы;
- предоставлены рекомендации по совершенствованию бизнес–процессов в организации и обоснована необходимость разработать информационно–аналитическую систему формирования отчетности;
- выбраны средства реализации, разработана и внедрена информационно–аналитическая система формирования отчетности;
- рассчитана экономическая эффективность внедренных отчетов.

Основной вывод данной работы в том, что внимательное изучение процессов в компании позволяют найти места, которые даже при незначительном изменении могут принести хорошую экономическую выгоду для предприятия.

Данная ИАС успешно внедрена в производство, о чем свидетельствует акт о внедрении (Приложение Г).

СПИСОК СОКРАЩЕНИЙ

ГСМ – горюче–смазочные материалы.

ДУТ – датчик уровня топлива.

Отдел ИА – отдел информационной аналитики.

Отдел РСПО – отдел разработки и сопровождения программного обеспечения.

ССМ – системы спутникового мониторинга.

ТС – транспортное средство.

ЭиРОЭ – эксплуатация и ремонт объектов энергохозяйства.

ЯП – язык программирования.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Коккоз М. М., Жумабай А. Р. Методика построения информационно-аналитических систем / Молодой ученый [Электронный ресурс]: moluch.ru — 2016. — Режим доступа: https://moluch.ru/archive/108/26272/
2. Александров И. Космическая радионавигационная система НАВСТАР (рус.) // Зарубежное военное обозрение. — М., 1995. — № 5. — С. 52—63.
3. Dan Cho. Space Tracker. The earliest satellite watchers' ideas led to GPS. Technology Review (2004-12-1)
4. Состав и состояние группировки GPS [Электронный ресурс]: http://www.sdcm.ru/ - Режим доступа: http://www.sdcm.ru/smglo/grupgps?site=extern&version=rus
5. Современные ГНСС. Основные характеристики систем навигации Информационный портал системы ГЛОНАСС. [Электронный ресурс] – Режим доступа: http://www.glonassgsm.ru
6. Постановление Правительства Российской Федерации от 25 августа 2008 г. N 641 г. Москва "Об оснащении транспортных, технических средств и систем аппаратурой спутниковой навигации ГЛОНАСС или ГЛОНАСС/GPS"
7. GPS Профи / Зияющие высоты ГЛОНАСС [Электронный ресурс]: http://gps-profi.ru/ - Режим доступа: http://gps-profi.ru/glonass03-2009.php
8. Всем по навигатору! А.Д. Кулагин / Журнал «За рулем» [Электронный ресурс]: https://www.zr.ru/ - Режим доступа: https://www.zr.ru/content/articles/242447-vsem_po_navigatoru/
9. Globalstar SmartOne — первый спутниковый трекер, поддерживаемый Wialon / Корпоративный Блог Gurtam [Электронный ресурс]: https://gurtam.com/ - Режим доступа: https://gurtam.com/ru/blog/first-satellite-gps-tracker-integrated-in-wialon
10. Лобанова М.Е. Информационная поддержка процессов транспортировки как инструмент управления грузоперевозками [Текст] / М.Е. Лобанова // Управление логистическими системами: глобальное мышление - эффективные решения (том 2). - Ростов н/Д : Изд.-полиграф. комплекс РГЭУ (РИНХ), 2014. - С. 141-144.
11. Топ-10 производителей оборудования систем для спутникового мониторинга / Системы спутникового мониторинга [Электронный ресурс]: https://ssm22.ru/ - Режим доступа: https://ssm22.ru/top-10-proizvoditelej-oborudovaniya-dlya-monitoringa-transporta/
12. Группа компаний «Техноком» [Электронный ресурс]: http://www.tk-nav.ru/
13. Официальный сайт компании Neomatica [Электронный ресурс]: http://neomatica.ru/
14. Сайт компании АРКСОМ / О компании [Электронный ресурс]: http://www.apkcom.com – Режим доступа: http://www.apkcom.com/about/

15. Выставка «Навитех» / Участники выставки 2015 года компании [Электронный ресурс]: http://www.navitech-expo.ru – Режим доступа: http://www.navitech-expo.ru/ru/history/2015/list2015/

16. Руководство пользователя Программы «AutoGRAPH 5 Pro» [Электронный ресурс] - Режим доступа: http://i.tk-chel.ru/docs/ru/AG.NET/USER_MANUAL_AG_5_PRO_231(full).pdf

17. Интернет-журнал «BYTE Россия» / Модель SaaS – в мире и в России [Электронный ресурс]: https://www.bytemag.ru – Режим доступа: https://www.bytemag.ru/articles/detail.php?ID=12825

18. Официальный сайт системы «СКАУТ» [Электронный ресурс]: Режим доступа: https://scout-gps.ru

19. Описание работы методов Autograph.NET [Электронный ресурс]: Режим доступа: http://wiki.tk-chel.ru/index.php/AutoGRAPH.NET_Service_Methods

20. Квоты использования ресурсов Google [Электронный ресурс]: Режим доступа: https://developers.google.com/apps-script/guides/services/quotas

21. Распеределенная система версий [Электронный ресурс]: Режим доступа:https://ru.wikipedia.org/wiki/Git

22. Описание библиотеки logging [Электронный ресурс]: Режим доступа: https://docs.python.org/3/library/logging.html

23. Google Sheets API v.4 Python Quickstart https://developers.google.com/sheets/api/quickstart/python

24. Метод append [Электронный ресурс]: Режим доступа: https://developers.google.com/sheets/api/reference/rest/v4/spreadsheets.values/append

Приложение А – листинг кода прототипа автоматического отчета

```
function menu() {
var SS =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/**/edit#gid=0');
var entries = [
{name: "Обновить таблицу бензовозов", functionName: "updateFuelTruckSheet" },
{ name: "Начать работу скрипта", functionName: "getOnlineDataFromAvt" }];
SS.addMenu("Бензовозы", entries);}

function NormalTime(y) {
x = new Date (y);
x.setTime(x.getTime()+(14*60*60*1000));
return x.toISOString().substring(0,19);}

function gelToken(){
var avtUrl = "http://****/ServiceJSON/";
var agMetodEnumDevices = "EnumDevices" + "?";
var agTokenandScheme = "AG-TOKEN=****&schemaID=****;
var urlDevices = avtUrl + agMetodEnumDevices + agTokenandScheme;}

function updateFuelTruckSheet() {
// переменная с таблицей, в которой происходит работа, открывается по URL
var SS =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/**/edit#gid=0');
// открываем сводную таблицу бензовозов
var ssFuel =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/***/edit#gid=308700385')
;
// в таблице забираем диапазон данных по именованному диапазону "Отчет_B_E"
var rangeInfoFuel = ssFuel.getRangeByName("Отчет_B_E").getValues();
// Заносим данные из именнованого диапазона на лист с именем "1"
SS.getSheetByName('1').getRange(2, 2, rangeInfoFuel.length,
rangeInfoFuel[0].length).setValues(rangeInfoFuel);
// Заносим названия заголовков на листе "1"
SS.getSheetByName('1').getRange(1, 1, 1, rangeInfoFuel[0].length + 4).setValues([['Дата
последнего забора данных', 'Имя', 'Гос.номер', 'Участок', 'Номер АВТ', 'ID', "Статус
выполнения", "Дата из GetOnlineInfoAll"]]);
// создаем переменную для Айди ТС
var IDs = []
// Создаем в эту переменную только номера ТС
var avtographSeries = SS.getSheetByName('1').getRange(2, 5,
SS.getSheetByName('1').getLastRow(), 1).getValues();
// Циклом заполняем переменную IDs словом "Нет"
for (var i = 0; i < avtographSeries.length; i++) {
IDs.push("Нет");}
// переменные для составления запроса к WEB Автографу
var avtUrl = "http://94.73.215.145:7771/ServiceJSON/";
var agMetodEnumDevices = "EnumDevices" + "?";
var agTokenandScheme = "AG-TOKEN=****&schemaID=****";
var urlDevices = avtUrl + agMetodEnumDevices + agTokenandScheme;
// в данную переменную заносим результат запроса EnumDevices к автографу, в формате JSON
var jsonDevices = JSON.parse(UrlFetchApp.fetch(urlDevices));
// в данную переменную заносим поле Items JSON запроса EnumDevices
var itemsFromJsonDevises = jsonDevices.Items;
// Данным циклом сравниваем номера из автографа и при нахождении забираем данные В
переменную Ids
for (i = 0; i < avtographSeries.length - 1; i++) {
for (var j = 0; j < itemsFromJsonDevises.length; j++) {
            if (Number(avtographSeries[i]) == itemsFromJsonDevises[j]['Serial']){
                    IDs[i] = itemsFromJsonDevises[j]['ID'];}}
SS.getSheetByName('1').getRange(i + 2, 6).setValue(IDs[i]);}}

function getOnlineDataFromAvt() {
var SS =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/**/edit#gid=0');
// узнаем номер последней строки
var lastRow = SS.getSheetByName('1').getLastRow();
// переменная со значениями, забираем от 2 строки до последней
var finallist = SS.getSheetByName('1').getRange(2, 1, lastRow - 1, 6).getValues();
```

```
// собираем запрос к сервису
var avtUrl = "http://****/ServiceJSON/";
var agTokenandScheme = "AG-TOKEN=****&schemaID=****";
var agMetodGetOnlineInfo = "GetOnlineInfo?";
// переменная, для номеров ID
var template = ' ';
// циклом заносим в template значения Ids бензовозов
for (var z = 0; z < finallist.length; z++) {
if (finallist[z][5] == 'Нет') continue;
template = template + finallist[z][5] + ',';}
var r;
// приводим строку в корректный вид
template = template.substring(1, template.length - 1);
// Собрали строку запроса
var urlGetOnlineInfo = avtUrl + agMetodGetOnlineInfo + agTokenandScheme + '&' + 'IDs=' +
template + '#';
// в данной переменной результат запроса в формате JSON
var jsonGetOnlineInfo = JSON.parse(UrlFetchApp.fetch(urlGetOnlineInfo));
// Основной цикл, кол-во итераций по кол-ву бензовозов
for (var i = 0; i < finallist.length; i++) {
var workSheet = SS.getSheetByName(String(finallist[1]) + " " + String(finallist[4]));
// если айдишника нет, или он пустой, переходим на следующий бензовоз
if (finallist[i][5] == "Нет" || jsonGetOnlineInfo[finallist[i][5]] == null ||
jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10) ==
SS.getSheetByName('1').getRange(i + 2, 8).getValue()) continue;
// Если дата последних данных из первого столбца не равен дате последних данных из
автографа
if (jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10) != finallist[i][0])
{
// заносим в переменную workSheet лист с именем бензовоза
      // если такой лист найден
if (workSheet != null) {
// выполняем код в оболочке try, чтобы при возникновении ошибок код продолжал работу
                    try {
r = getJSONfromAVT(finallist[i],
jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));
if (r[finallist[i][5]] == null) throw new SyntaxError("Пустой JSON");
                           getNewDataFromAVT(workSheet, r, finallist[i]);
// возвращаемся на лист '1' и меняем дату последнего забора данных
SS.getSheetByName('1').getRange(i+
2,1).setValue(jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));
// пишем результат по данному бензовозу
                           SS.getSheetByName('1').getRange(i + 2, 7).setValue("Успех");}
// Если по ходу выполнения двух функций была какая-то ошибка, то
                    catch (err) {
                           // пишем текст ошибки
                           SS.getSheetByName('1').getRange(i + 2, 7).setValue(err);
SS.getSheetByName('1').getRange(i + 2,
9).setValue(jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));}}
// если в переменной workSheet нет ничего (лист не создавался по бензовозу)
             if (workSheet == null) {
try {
r = getJSONfromAVT(finallist[i],
jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));
if (r[finallist[i][5]] == null) throw new SyntaxError("Пустой JSON");
getNewDataFromAVT(createSheetforNewFuelTruck(SS, finallist[i]), r, finallist[i]);
SS.getSheetByName('1').getRange(i + 2,
1).setValue(jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));
                           SS.getSheetByName('1').getRange(i + 2, 7).setValue("Успех");}
catch (err) {
                           SS.getSheetByName('1').getRange(i + 2, 7).setValue(err);
SS.getSheetByName('1').getRange(i + 2,
8).setValue(jsonGetOnlineInfo[finallist[i][5]]["_LastData"].substring(0, 10));}}}}}

function getJSONfromAVT(TC, SDate) {
// собираем запрос GetStage
var avtUrl = "http://****/ServiceJSON/";
var agMetodGetStage = "GetStage" + "?";
var agTokenandScheme = "AG-TOKEN=****&schemaID=****&";
// Номер ТС
var idTC = "IDs=" + TC[5] + "&";
```

```
// Дата начала
var sd = "SD=" + TC[0].substring(0, 4) + TC[0].substring(5, 7) + TC[0].substring(8, 10) +
"-0000" + "&";
// Дата конца
var ed = "ED=" + SDate.substring(0, 4) + SDate.substring(5, 7) + SDate.substring(8, 10) +
"-0000" + "&";
// По какому отрезку забираем параметры
var stageName = "stageName=Tank3&";
// забираемые параметры
var avtographTripParams = 'tripParams=' + 'FirstLocation,DateTime First,DateTime
Last,Tank3FuelUpVol Diff,Tank3FlowmeterDur Last,Tank3FlowmeterCard Last';
// собираем строку запроса
var urlGetStage = avtUrl + agMetodGetStage + agTokenandScheme +
idTC + sd + ed + stageName + avtographTripParams + "#";
// заносим JSON
var jsonGetStage = JSON.parse(UrlFetchApp.fetch(urlGetStage));
// в переменной результат запроса к сервису в формате JSON
// возвращаем запрос
return jsonGetStage}

function getNewDataFromAVT(workSheet, jsonGetStage, TC) {
var i = 0;
// Переменная для занесения корректных данных, которые уже отсортированы и ждут занесения
на лист
var finalData = [];
// получаем номер последней строки
var lastrow = workSheet.getLastRow();
// пытаемся изменить цвет всех строк, если не получается, просто идем дальше, тк то
первое занесение данных на лист
try {
workSheet.getRange(4, 1, lastrow - 3, 3).setBackground('yellow');}
// но на всякий случай заносим в лог ошибку
catch (err) { Logger.log(err);}
// цикл, который заносит данные в finalData
for (i = 0; i < jsonGetStage[TC[5]]["Items"].length; i++) {
// если заправка нулевая
if (jsonGetStage[TC[5]]["Items"][i]["Values"][0] ==
jsonGetStage[TC[5]]["Items"][i]["Values"][1]) {
a = NormalTime(jsonGetStage[TC[5]]["Items"][i]["Values"][0]);
            Logger.log('a' + a);
b = NormalTime(jsonGetStage[TC[5]]["Items"][i]["Values"][1]);
            finalData[i] = [jsonGetStage[TC[5]]["Items"][i]["Values"][2],
a.substring(0, 10) + ' ' + a.substring(11),  b.substring(0, 10) + ' ' + b.substring(11),
" ",jsonGetStage[TC[5]]["Items"][i]["Values"][3],null,"=" + "E" + String(lastrow + 1 + i)
+ '-F' + String(lastrow + 1 + i),'0:00:00',''''];}
       // если заправка ненулевая
else {
a = NormalTime(jsonGetStage[TC[5]]["Items"][i]["Values"][0]);
            b = NormalTime(jsonGetStage[TC[5]]["Items"][i]["Values"][1]);
finalData[i] = [jsonGetStage[TC[5]]["Items"][i]["Values"][2],a.substring(0, 10) + ' ' +
a.substring(11),b.substring(0, 10) + ' ' + b.substring(11),"
",jsonGetStage[TC[5]]["Items"][i]["Values"][3],null,"=" + "E" + String(lastrow + 1 + i) +
'-F' + String(lastrow + 1 +
i),sonGetStage[TC[5]]["Items"][i]["Values"][4],String(jsonGetStage[TC[5]]["Items"][i]["Va
lues"][5])];}}


// заносим значения на лист
workSheet.getRange(workSheet.getLastRow() + 1, 1, finalData.length,
9).setValues(finalData);
// конечное форматирование листа
var temp = workSheet.getRange('A1:L1').getValue();
workSheet.getRange('A1:L1').breakApart().clear();
workSheet.autoResizeColumns(1, 12);
workSheet.getRange('A1:L1').merge().setValue(temp).setFontWeight('bold').setFontSize(16).
setHorizontalAlignment("center");
lastrow = workSheet.getLastRow();
workSheet.getRange(4, 5, lastrow, 3).setNumberFormat('0.00');
workSheet.getRange(4, 10, lastrow, 2).setNumberFormat('0.00');
workSheet.getRange(1, 1, lastrow, 12).setBorder(true, true, true, true, true, true);}
```

```
function createSheetforNewFuelTruck(SS, finallist) {
try {
var workSheet = SS.insertSheet().setName(String(finallist[1]) + " " +
String(finallist[4]));}
catch (err) {
SS.deleteActiveSheet();
var workSheet = SS.getSheetByName(String(finallist[1]) + " " + String(finallist[4]));}
workSheet.getRange('A2:A3').merge().setValue('Место').setFontWeight('bold').setFontSize(1
2).setHorizontalAlignment("center").setVerticalAlignment("middle");
workSheet.getRange('B2:C2').merge().setValue('Время').setFontWeight('bold').setFontSize(1
2).setHorizontalAlignment("center");
workSheet.getRange('B3').setValue('Начало').setFontWeight('bold').setFontSize(12).setHori
zontalAlignment("center");
workSheet.getRange('C3').setValue('Конец').setFontWeight('bold').setFontSize(12).setHoriz
ontalAlignment("center");
workSheet.getRange('D2:D3').merge().setValue('Заправка цистерны АТЗ,
л.').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center").setVerticalAl
ignment("middle");
workSheet.getRange('E2:F2').merge().setValue('Заправки
ТС').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center");
workSheet.getRange('E3').setValue('ПОРТ-3,
л.').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center");
workSheet.getRange('F3').setValue('Ведомость,
л.').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center");
workSheet.getRange('G2:G3').merge().setValue('Разница,\nл.').setFontWeight('bold').setFon
tSize(12).setHorizontalAlignment("center").setVerticalAlignment("middle");
workSheet.getRange('H2:H3').merge().setValue('Длительность').setFontWeight('bold').setFon
tSize(12).setHorizontalAlignment("center").setVerticalAlignment("middle");
workSheet.getRange('I2:I3').merge().setValue('Водитель').setFontWeight('bold').setFontSiz
e(12).setHorizontalAlignment("center").setVerticalAlignment("middle");
workSheet.getRange('J2:J3').merge().setValue('Затир, ППО,
л.').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center").setVerticalAl
ignment("middle");
workSheet.getRange('K2:K3').merge().setValue('Затир,
ведомость,\nл.').setFontWeight('bold').setFontSize(12).setHorizontalAlignment("center").s
etVerticalAlignment("middle");
workSheet.getRange('L2:L3').merge().setValue('Комментарий').setFontWeight('bold').setFont
Size(12).setHorizontalAlignment("center").setVerticalAlignment("middle");
workSheet.autoResizeColumns(1, 12);
workSheet.getRange('A1:L1').merge().setValue('Отчет о работе ' + finallist[1] + ', Гос
номер ' + finallist[2] + ', закрепленный за уч. ' +
finallist[3]).setFontWeight('bold').setFontSize(16).setHorizontalAlignment("center");
workSheet.setFrozenRows(3);
return workSheet;}

function updateLogFile() {
var logFile = DocumentApp.openById('****');
var lastRow =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/****/edit#gid=0').getShe
etByName('1').getLastRow();
var logsFromSS =
SpreadsheetApp.openByUrl('https://docs.google.com/spreadsheets/d/****/edit#gid=0').getShe
etByName('1').getRange(2, 7, lastRow, 2).getValues();}

function delAll() {
var as = SpreadsheetApp.getActiveSpreadsheet();
as.getSheets().forEach(function (s) {
var sname = s.getSheetName();
if (sname != '1') as.deleteSheet(s);})}
```

Приложение Б – листинг кода AutographAPI.py

```
import datetime
import requests
import json
import logging
from Constants import *
import re
import time
# Запускаем логирование
logger = logging.getLogger(__name__)
logger.setLevel(logging.WARNING)
# Указываем файл для записи лога
dateTag = datetime.datetime.now().strftime("%Y-%m-%d")
fh = logging.FileHandler("Log %s.log" % dateTag)
# Указываем формат записи сообщения
formatter = logging.Formatter(u'[%(asctime)s] %(levelname)-9s %(filename)-
15s[LINE:%(lineno)-4s]# %(message)s')
# Добавляем формат сообщения в файл лога
fh.setFormatter(formatter)
# Добавляем файл записи
logger.addHandler(fh)
class DateClass():
    """ класс DateClass призван проинициализировать будующий url ip-адресом,
портом, и форматом хранения времени
        также, управляет переводом времени из формата datetime в нужный нам формат
для вызова по url-ссылке """

    HTTP_PART = "http://"

    def __init__(self, ip_adress: str = '94.73.215.145', port: str = '57772',
time_format: str = '%Y-%m-%d %H:%M:%S', session = None):
        self.session = session or requests.Session()
        self.ip_adress = ip_adress
        self.port = port
        self.time_format = time_format

    def DomainPart(self) -> str:
        return DateClass.HTTP_PART + self.ip_adress + ':' + self.port

    def DatetimeToURL(self, date: datetime.datetime) -> str:
        return datetime.datetime.strftime(date, self.time_format)

    def date_to_date(self, date: str, old_format: str, new_format: str) -> str:
        return datetime.datetime.strptime(date, old_format).strftime(new_format)

    def GetJsonDataByUrl(self, url: str, auth: tuple = ('Analyst', 'Aa123456')) ->
object:
        return json.loads(self.session.get(url, auth=auth).text)

    def __str__(self):
       return '{0}{1}:{2}'.format(DateClass.HTTP_PART, self.ip_adress, self.port)

class NetServiceUrl(DateClass):

    UserName = 'vreports'
    Password = '****'
    token = "****"  # vreport 45
```

```
    # token = "****" # mve001 209

    def __init__(self, ip_adress: str = '192.168.1.45', port: str = '8300',
time_format: str = '%Y%m%d-%H%M%S', schema_name: str = 'Scheme'):
        super().__init__(ip_adress, port, time_format)
        self.schema_name = schema_name

    def Login(self, user_name: str = UserName, password: str = Password) -> str:
        auth = {'UserName' : user_name, 'Password' : password}
        self.token = requests.post('{0}/Login'.format(self.DomainPart()), json =
auth).text[1:-1]
        return self.token

    def GetStage(self, schema_name: str, uid: str, SD: datetime.datetime, ED:
datetime.datetime, stage_name: str, trip_params: str ='*', trip_total_params: str =
'*', auth: bool = True) -> str:
        url = self.DomainPart() + '/GetStage?'
        url += 'session={0}'.format(self.token) if auth == True else ''
        url +=
'&schemaID={0}&IDs={1}&SD={2}&ED={3}&stageName={4}'.format(schema_name, uid,
self.DatetimeToURL(SD), self.DatetimeToURL(ED), stage_name)
        url += '&tripParams={0}'.format(trip_params) if trip_params != '*' else ''
        url += '&tripTotalParams={0}'.format(trip_total_params) if
trip_total_params != '*' else ''
        return url

    def GetTripTables(self, schema_name: str, uid: str, SD: datetime.datetime, ED:
datetime.datetime, online_params: str, trip_splitter_index: int = -1, auth: bool =
True) -> str:
        url = self.DomainPart() + '/GetTripTables?'
        url += 'session={0}'.format(self.token) if auth == True else ''
        url +=
'&schemaID={0}&IDs={1}&SD={2}&ED={3}&onlineParams={4}'.format(schema_name, uid,
self.DatetimeToURL(SD), self.DatetimeToURL(ED), online_params)
        url += '&tripSplitterIndex={0}'.format(trip_splitter_index) if
trip_splitter_index != -1 else ''
        return url

    def GetOnlineInfo(self, schema_name: str, uid: str, final_params: str = '*',
auth: bool = True) -> str:
        url = self.DomainPart() + '/GetOnlineInfo?'
        url += 'session={0}'.format(self.token) if auth == True else ''
        url += '&schemaID={0}&IDs={1}'.format(schema_name, uid)
        url += '&finalParams={0}'.format(final_params) if final_params != '*' else
''
        return url

    def GetOnlineInfoAll(self, schema_name: str, final_params: str = '*', auth:
bool = True) -> str:
        url = self.DomainPart() + '/GetOnlineInfoAll?'
        url += 'session={0}'.format(self.token) if auth == True else ''
        url += '&schemaID={0}'.format(schema_name)
        url += '&finalParams={0}'.format(final_params) if final_params != '*' else
''
        return url

    def EnumDevices(self, schema_name: str, auth: bool = True) -> str:
        url = self.DomainPart() + '/EnumDevices?'
```

```
        url += 'session={0}'.format(self.token) if auth == True else ''
        url += '&schemaID={0}'.format(schema_name)
        return url

    def GetTripTotal(self, schema_name, uid: str, SD: datetime.datetime, ED:
datetime.datetime, trip_Splitter_Index: int = -1, trip_params: str = '*',
trip_total_params: str = '*', auth: bool = True) -> str:
        """ Функция возвращает URL ссылку на метод GetTripsTotal к NET Service
        Принимает: uid Транспортного средства, SD/ED - время начала и конца
рассматриваемого отрезка,
        stage_name - имя параметра, по которому NET Service разобьет данные на
отрезки,
        trip_params, total_trip_params - ???
        """
        url = self.DomainPart() + '/GetTripsTotal?'
        url += 'session={0}'.format(self.token) if auth == True else ''
        url += '&schemaID={0}&IDs={1}&SD={2}&ED={3}'.format(schema_name, uid,
self.DatetimeToURL(SD), self.DatetimeToURL(ED))
        url += '&tripSplitterIndex={0}'.format(trip_Splitter_Index) if
trip_Splitter_Index != '*' else ''
        return url

    def GetJsonDataByUrlTries(self,url):
        res = None
        tries= 1
        while res is None:
            try:
                data = requests.get(url).text
                res = json.loads(data)
            except Exception:
                pass
            else:
                logger.info('JSON получен успешно по запросу {0}'.format(url))
                return res
            finally:
                if res is None:
                    self.WriteError('Получить JSON не удалось, попытка {0}, запрос
{1}'.format(tries, url))
                    tries += 1
                    time.sleep(30)

    def GetJsonDataByUrl(self, url: str) -> object:
        res = self.GetJsonDataByUrlTries(url)
        return res

    def FormatDate(self,date, format_: str = '%Y-%m-%d', greenvich: int = 1):
        '''Метод принимает кол-во милисекунд и преобразует в дату в указанном
формате
        Параметр greenvich - при 0 нет смещения по времени, при 1 добавляется 7
часов, при 2 убирается 7 часов
        По умолчанию стоит 1'''
        microseconds = re.search(r'[0-9]*\d+' , date)[0]
        if greenvich == 1:
            seconds = int(microseconds) / 1000 + 7*60*60
        if greenvich == 0:
            seconds = int(microseconds) / 1000
        if greenvich == 2:
            seconds = int(microseconds) / 1000 - 7*60*60
```

```
        zero_date = datetime.datetime(1970, 1, 1, tzinfo=datetime.timezone.utc)
        date_time = zero_date + datetime.timedelta(seconds = seconds)
        return datetime.datetime.strftime(date_time, format_)

    def FormatDurToInt(self, val: str, dg: int = 1):
        '''Метод переводит строку продолжительности вида 00:00:00 в числовое
представление
        val - значение продолжительности
        dg - степень округления (по умолчанию до 1 знака)'''
        try:
            val = val.split(':')
            res = int(val[0]) + (int(val[1]) / 60) + (int(val[2]) / 3600)
            res = round(res, dg)
            return res
        except Exception as e:
            logger.error(f'не удалось перевести продолжительность в числовое
представление:{type(e)} {e}')
            return val
```

Приложение В – листинг кода GoogleSheetAPI.py

```
class GoogleSheetsAPI():
    """Класс для работы с методами Google Sheets API v4,
    актуальная версия API для работы с таблицами."""
    credentials: ServiceAccountCredentials

    def __init__(self,
        credentials_file: str = 'C:\TCReports\\automatic-reports-to-
ssm\TankersReport\Testing1-e87cb0cf5ecc.json',
        api_type = 'sheets', api_version = 'v4'):
        try:
            self.credentials =
ServiceAccountCredentials.from_json_keyfile_name(credentials_file,
['https://www.googleapis.com/auth/spreadsheets',
'https://www.googleapis.com/auth/drive'])
            httpAuth = self.credentials.authorize(httplib2.Http())
            self.service = googleapiclient.discovery.build(api_type, api_version,
http = httpAuth, cache_discovery = False)
            logger.info('Подключение к Google Spreadsheet API успешно')
        except Exception as e:
            logger.error('Подключение к Google Spreadsheet API вернула ошибку:
{0}'.format(e))

    def CreateNewSheet(self, sheet_name: str, spreadsheet_id: int, sheet_id: int =
None,
        columns: int = 14, rows: int = 100,
        frozen_columns: int = 0, frozen_rows: int = 0):
        """ Функция создает новый лист.
        sheet_name - название нового листа;
        spreadsheet_id - ID таблицы, где нужно создать новый лист;
        sheet_id - id нового листа (по умолчанию случайный номер);
        columns - количество столбцов на новом листе (по умолчанию 14);
        frozen_columns - количество закрепленных столбцов (по умолчанию - 0);
        rows - количество строк на новом листе (по умолчанию 100);
        frozen_rows - количество закрепленных столбцов (по умолчанию - 0).

        Возвращает: Возвращает JSON - ответ от Google API,
        При возврате None создать лист не получилось """
        try:
            body = {
            "includeSpreadsheetInResponse": True,
            "requests": [{
                "addSheet": {
                    "properties": {
                        "title": sheet_name,
                        "gridProperties": {
                            "columnCount": columns,
                            "rowCount": rows,
                            "frozenColumnCount": frozen_columns,
                            "frozenRowCount": frozen_rows}
                        }}}]}
            if sheet_id != None:
                body['requests'][0]['addSheet']['properties']['sheetId'] = sheet_id
            res =
self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body).execute()
            logger.info('Лист успешно создан')
```

```
            return res
        except Exception as e:
            logger.error('При попытке создания листа произошла ошибка {0}
{1}'.format(type(e),e))

    def AppendNewRows(self, spreadsheet_id: int, sheet_name: str, range: str,
values: list, insert_option: str = "OVERWRITE"):
        """Метод добавляет новые строки на лист.
        spreadsheet_id - ID таблицы, куда нужно добавить строки;
        sheet_name - имя листа, куда нужно добавить строки;
        range - ячейка в нотации A1, с которой нужно записать данные (левая верхняя
ячейка);
        values - значения, в формате [[A1, B1, C1, ...], [A2, B2, C2], ...];
        insert_option - INSERT_ROWS добавляет новые строки по кол-ву добавленных
строк на лист, OVERWRITE (по умолчанию) не добавляет новые строки без
необходимости.

         Возвращает результат изменения: диапазон {'updates' : {"updatedRange"}}
        """
        body = {"majorDimension": "ROWS",
                "values": values
                }
        try:
            res =
self.service.spreadsheets().values().append(spreadsheetId=spreadsheet_id, range =
'{0}!{1}'.format(sheet_name, range), body= body,

valueInputOption="USER_ENTERED", insertDataOption = insert_option).execute()
            logger.info('Данные занесены успешно')
            logger.debug('{0}'.format(res))
            return res
        except Exception as e:
            logger.error('ошибка при занесении строк {0} {1}'.format(type(e), e))
            pass

    def GetSheetsNameFromSpreadsheet(self, spreadsheet_id: str):
        '''Метод возвращает список объектов, Поля которого: title - имя листа,
        ID листа (sheetId) и порядковый номер листа (index).
        Принимает: spreadsheet_id - id таблицы, с которой требуется забрать
информацию о листах'''
        time.sleep(10)
        data =
self.service.spreadsheets().get(spreadsheetId=spreadsheet_id).execute()
        list_of_sheets_name = []
        for sheet in data['sheets']:
            title = sheet['properties']['title']
            sheetId = sheet['properties']['sheetId']
            index = sheet['properties']['index']
            list_of_sheets_name.append({'title' : title, 'sheetId' : sheetId,
'index': index})
        return list_of_sheets_name

    def CopySheet(self, spreadsheet_id: str, destination_spreadsheet_id: str,
sheet_id = None, sheet_name = None):
        '''Метод копирует указанный лист по ID или названию листа
        Возвращает объект с данными нового листа или None при ошибке
        При отсутствии в методе параметра ID или названия листа (и одновременном их
наличии) выдаст исключение
```

```
        Возвращает: { 'gridProperties': {'columnCount' , 'rowCount'}
          'index': индекс листа в новой таблице,
          'sheetId': id листа в новой таблице,
          'sheetType': тип листа (GRID или OBJECT),
          'title': название листа в новой таблице }
        Принимает: spreadsheet_id - ID таблицы, из которой требуется получить лист
        destination_spreadsheet_id - ID таблицы, в которую требуется скопировать
лист
        sheet_id - ID листа (default None)
        sheet_name - название листа (default None)
        '''
        try:
            if sheet_id == sheet_name or (sheet_id and sheet_name is not None):
                raise TypeError('Некорректное значения аргументов sheet_id /
sheet_name:')
            if sheet_name is not None:
                res = False
                list_sheets = self.GetSheetsNameFromSpreadsheet(spreadsheet_id)
                for sheet in list_sheets:
                    if sheet['title'] == sheet_name:
                        sheet_id = sheet["sheetId"]
                        res = True
                if res is False:
                    raise Exception('В таблице с ID {0} нет листа с названием
{1}'.format(spreadsheet_id, sheet_name))
            if sheet_id is not None:
                res = self.service.spreadsheets().sheets().copyTo( spreadsheetId =
spreadsheet_id,
                                                    sheetId = sheet_id,
                                                    body = { "destinationSpreadsheetId":
destination_spreadsheet_id }).execute()
                logger.info('Копирование произошло успешно')
                return res
        except Exception as e:
            logger.error('Метод копирования листа вернула ошибку: {0}'.format(e))

    def RenameSheet(self, spreadsheet_id, new_sheet_name, sheet_id = None,
old_sheet_name = None):
        """
        Метод меняет название листа в таблице по ID или старому названию.
        Возвращает объект с данными нового листа или None при ошибке.
        При отсутствии в методе параметра ID или названия листа (и одновременном их
наличии) выдаст исключение.

        Принимает: spreadsheet_id - ID таблицы;
        new_sheet_name - новое название листа;
        sheet_id - ID листа (default None);
        old_sheet_name - старое название листа (default None).
        """
        try:
            if sheet_id == old_sheet_name or (sheet_id and old_sheet_name is not
None):
                raise TypeError('Некорректное значения аргументов sheet_id /
old_sheet_name:')
            if old_sheet_name is not None:
                res = False
                list_sheets = self.GetSheetsNameFromSpreadsheet(spreadsheet_id)
                for sheet in list_sheets:
```

```
                    if sheet['title'] == old_sheet_name:
                        sheet_id = sheet["sheetId"]
                        res = True
                if res is False:
                    raise Exception('В таблице с ID {0} нет листа с названием
{1}'.format(spreadsheet_id, old_sheet_name))
            if sheet_id is not None:
                    body_for_rename_sheet = {"requests":[
                                                {"updateSheetProperties":
                                                    {"properties":
                                                        {"title": new_sheet_name,
                                                        "sheetId": sheet_id},
                                                        "fields": "title"}}]}
                    res = self.service.spreadsheets().batchUpdate(spreadsheetId =
spreadsheet_id, body = body_for_rename_sheet).execute()
                    logger.info('Название листа изменено на
{1}'.format(new_sheet_name))
                    return res
        except Exception as e:
            logger.error('Метод переименования листа вернул ошибку: {0},
{1}'.format(type(e),e))

    def GetValues(self, spreadsheet_id, sheet_name, range_, major_dimension =
'ROWS'):
        """Метод возвращает объект со значениями строк в указанном диапазоне
нотации A1 и указанном листе
        'majorDimension' - как организованы значения (ROWS или COLUMNS)
        'range' - диапазон забранных значений
        'values' - сами данные
        """
        try:
            logger.info("Получение из листа {0} значений в диапазоне
{1}".format(sheet_name,range_))
            res = self.service.spreadsheets().values().get(spreadsheetId =
spreadsheet_id, range = '{0}!{1}'.format(sheet_name,range_), majorDimension =
major_dimension).execute()
            logger.debug('Данные получены успешно {0}'.format(res))
            return res
        except Exception as e:
            logger.error("При попытке получить значения в диапазоне {0} листа {1} в
таблице {2} возникла ошибка: {3}".format(range_,sheet_name,spreadsheet_id,e))

    def UpdateValues(self, spreadsheet_id, sheet_name, range_, values):
        """Метод заносит данные в таблицу по выбранному диапазону, перезаписывая их
        Возвращает: объект с информацией об изменениях
        Принимает:
        spreadsheet_id - ID таблицы,
        sheet_name - имя листа,
        range_ - диапазон ячеек, в которые требуется добавить данные,
        values - список (строки), содержащий списки(значения столбцов)"""
        try:
            logger.info("Обновление данных на листе {0} в диапазоне
{1}".format(sheet_name,range_))
            body_ = {"data":[
                        {"majorDimension": "ROWS",
                        "range": '{0}!{1}'.format(sheet_name,range_),
                        "values": values}],
                    "valueInputOption": "USER_ENTERED"}
```

```
            res = self.service.spreadsheets().values().batchUpdate(spreadsheetId =
spreadsheet_id,  body = body_).execute()
            logger.info('Данные изменены успешно {0}'.format(res))
            return res
        except Exception as e:
            logger.error("При попытке изменить значения в диапазоне {0} листа {1} в
таблице {2} возникла ошибка: {3}".format(range_,sheet_name,spreadsheet_id,e))

    def GetMerges(self, spreadsheet_id: str, sheet_name):
        """ Метод возращает объект с информацией об объединенных ячейках выбранного
листа """
        metadata = self.service.spreadsheets().get(spreadsheetId =
spreadsheet_id).execute()
        sheets = metadata['sheets']
        for sheet in sheets:
            if sheet['properties']["title"] == sheet_name:
                return sheet["merges"]

    def SetMerges(self, spreadsheet_id, sheet_id, merges_list):
        '''Метод задает объединение ячеек по указанному объекту (Получить можно
GetMerges)'''
        try:
            body = {'requests': []}
            for merge in merges_list:
                merge["sheetId"] = sheet_id
                merge_cell = {'mergeCells': {'range':{}}}
                merge_cell['mergeCells']['range'] = merge
                body['requests'].append(merge_cell)
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body).execute()
        except Exception as e:
            logger.error("При попытке объединить ячейки возникла ошибка: {0}
{1}".format(type(e),e))

    def SetBordersAll(self, spreadsheet_id, sheet_id, range_, width: int = 1, style
= 'SOLID'):
        '''Метод рисует все границы ячеек в выбранном диапазоне, диапазон
реализован листом
        [номер столбца начала, номер строки начала, номер столбца конца, номер
строки конца], в числах'''
        body = {"requests": [
                    {"updateBorders": {
                        "innerHorizontal": {
                            "style": style,
                            "width": width},
                        "innerVertical": {
                            "style": style,
                            "width": width},
                        "top": {
                            "style": style,
                            "width": width},
                        "bottom": {
                            "style": style,
                            "width": width},
                        "left": {
                            "style": style,
                            "width": width},
                        "right": {
```

```
                            "style": style,
                            "width": width},
                        "range": {
                            "startColumnIndex": range_[0],
                            "startRowIndex": range_[1],
                            "endColumnIndex": range_[2],
                            "endRowIndex": range_[3],
                            "sheetId": sheet_id}
                    }},




                    ]}
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body).execute()
            logger.info('Границы на листе нарисованы успешно')
        except Exception as e:
            logger.error('При попытке нарисовать границы произошла ошибка: {0},
{1}'.format(type(e), e))

    def SetHeaderFormat(self, spreadsheet_id, sheet_id, range_, width: int = 1,
style = 'SOLID', bold: bool = True):
        """Метод оформляет шапку отчета, выравнивает значения по центру, рисует
границу и жирный шрифт
        диапазон шапки реализован листом
        [номер столбца начала, номер строки начала, номер столбца конца, номер
строки конца]"""
        body_ = {"requests":[
                    {"repeatCell":
                        {"cell":
                            {"userEnteredFormat":
                                {"borders":
                                    {"bottom":
                                        {"style": style,
                                        "width": width},
                                    "left":
                                        {"style": style,
                                        "width": width},
                                    "right":
                                        {"style": style,
                                        "width": width},
                                    "top":
                                        {"style": style,
                                        "width": width}},

                                    "textFormat":
                                        {"bold": bold},
                                    "horizontalAlignment": "CENTER",
                                    "verticalAlignment": "MIDDLE"}},
                            "range": {
                                "startColumnIndex": range_[0],
                                "startRowIndex": range_[1],
                                "endColumnIndex": range_[2],
                                "endRowIndex": range_[3],
                                "sheetId": sheet_id},
                            "fields": "userEnteredFormat"}}]}
```

```
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.info('Формат шапки отчета установлен')
        except Exception as e:
            logger.error('При попытке форматировании шапки отчета возникла ошибка:
{0}, {1}'.format(type(e), e))


    def SetBoldText(self,spreadsheet_id, sheet_id, range_):
        '''Метод делает полужирный шрифт в выбранном диапазоне'''
        body_ = {"requests":[
                    {"repeatCell":
                        {"cell":
                            {"userEnteredFormat": {
                                "textFormat": {"bold": True},
                                "horizontalAlignment": "CENTER",
                                "verticalAlignment": "MIDDLE"}
                            },
                            "range": {
                                "startColumnIndex": range_[0],
                                "startRowIndex": range_[1],
                                "endColumnIndex": range_[2],
                                "endRowIndex": range_[3],
                                "sheetId": sheet_id},
                            "fields": "userEnteredFormat.textFormat"}}]}
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.info('Текст ожирнел')
        except Exception as e:
            logger.error('При попытке ожирнить текст возникла ошибка: {0},
{1}'.format(type(e), e))


    def SetUsualText(self,spreadsheet_id, sheet_id, range_):
        '''Метод делает обычный шрифт в выбранном диапазоне'''
        body_ = {"requests":[
                    {"repeatCell":
                        {"cell":
                            {"userEnteredFormat": {
                                "textFormat": {"bold": False},
                                "horizontalAlignment": "CENTER",
                                "verticalAlignment": "MIDDLE"}
                            },
                            "range": {
                                "startColumnIndex": range_[0],
                                "startRowIndex": range_[1],
                                "endColumnIndex": range_[2],
                                "endRowIndex": range_[3],
                                "sheetId": sheet_id},
                            "fields": "userEnteredFormat.textFormat"}}]}
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.info('Текст ожирнел :) ')
        except Exception as e:
```

```
            logger.error('При попытке обычить текст возникла ошибка: {0},
{1}'.format(type(e), e))

    def SetNumberFormat(self,spreadsheet_id, sheet_id, range_ : list, type_ =
'TIME', pattern = '[h]:mm:ss'):
        '''Метод задает формам чисел в выбранном диапазоне, по умолчанию
устанавливет продолжительность'''
        body_ = {"requests":[
                    {'repeatCell': {
                        'range': {
                            "startColumnIndex": range_[0],
                            "startRowIndex": range_[1],
                            "endColumnIndex": range_[2],
                            "endRowIndex": range_[3],
                            "sheetId": sheet_id
                        },
                        'cell': {
                            'userEnteredFormat': {
                                'numberFormat': {
                                    'type': type_,
                                    'pattern': pattern,
                                },
                            },
                        },
                        'fields': 'userEnteredFormat.numberFormat'}}]}
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.info('Формат чисел задан :) ')
        except Exception as e:
            logger.error('При попытке форматировать числа возникла ошибка: {0},
{1}'.format(type(e), e))

    def DeleteRows(self, spreadsheet_id, sheet_id, start, end):
        '''Метод удаляет строки на листе'''
        body_ = {"requests":[
                    {"deleteDimension":
                        {"range": {
                            "sheetId": sheet_id,
                            "dimension": "ROWS",
                            "startIndex": str(start),
                            "endIndex": str(end)}}}]}
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.debug('строки удалены')
        except Exception as e:
            logger.error('При попытке удаления строки возникла ошибка: {0},
{1}'.format(type(e), e))

    def UpdateBackgroundColor(self, spreadsheet_id, sheet_id, cells_range, red :
int = 0, green : int = 0, blue : int = 0):
        if red != 0:
            red = red / 255
        if green != 0:
            green = green / 255
        if blue != 0:
            blue = blue / 255
```

```
        body_ ={"requests": [
                    {"repeatCell":
                        {"cell":
                            {"userEnteredFormat":
                                {"backgroundColor": {
                                    "blue": blue,
                                    "green": green,
                                    "red": red
                                    }
                                }
                            },
                        "range": {
                            "sheetId": sheet_id,
                            "startRowIndex": cells_range[1],
                            "startColumnIndex": cells_range[0],
                            "endRowIndex": cells_range[3],
                            "endColumnIndex": cells_range[2]},
                        "fields": "userEnteredFormat(backgroundColor)"
                        }
                    }]
                }
        try:
            self.service.spreadsheets().batchUpdate(spreadsheetId=spreadsheet_id,
body=body_).execute()
            logger.debug('цвет ячеек изменен')
        except Exception as e:
            logger.error('При попытке изменить цвет ячеек возникла ошибка: {0},
{1}'.format(type(e), e))



    def ToGridRange(self,cells_range):
        '''Метод возвращает объект с диапазоном ячеек, конвертированный из нотации
A1'''
        start_cell = None
        end_cell = None
        try:
            start_cell, end_cell = cells_range.split(":")[0:2]
        except ValueError:
            start_cell = cells_range.split(":")[0]
        cells_range = {}
        rangeAZ = range(ord('A'), ord('Z') + 1)
        try:
            if ord(start_cell[0]) in rangeAZ:
                cells_range["startColumnIndex"] = ord(start_cell[0]) - ord('A')
                start_cell = start_cell[1:]
            if ord(end_cell[0]) in rangeAZ:
                cells_range["endColumnIndex"] = ord(end_cell[0]) - ord('A') + 1
                end_cell = end_cell[1:]
        except Exception:
            pass
        try:
            if len(start_cell) > 0:
                cells_range["startRowIndex"] = int(start_cell) - 1
            if len(end_cell) > 0:
                cells_range["endRowIndex"] = int(end_cell)
        except Exception:
            pass
```

```
        return cells_range

    def ToA1Range(self,cells_range):
        '''Метод возвращает диапазон в нотации A1'''
        A1 = ['','',':','','']
        try:
            A1[0] = chr(ord('A') + cells_range['startColumnIndex'])
        except:
            pass
        try:
            A1[3] = chr(ord('A') + int(cells_range['endColumnIndex'])-1)
        except:
            pass
        try:
            A1[1] = str(int(cells_range['startRowIndex']) + 1 )
        except:
            pass
        try:
            A1[4] = str(cells_range['endRowIndex'])
        except:
            pass

        return ''.join(A1)

    def intToA1(self, number: int):
        '''метод переводит номер столбца в буквенное представление'''
        result = ""
        A = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        while number > 0:
            reminder = number % 26
            result += A[reminder - 1]
            number = number // 26 - (1 if reminder == 0 else 0)
        return result[::-1]

    def A1toInt(self, A: str):
        '''метод переводит буквенное представление в номер столбца'''
        a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        number = 0
        for index, i in enumerate(reversed(A)):
            number += (a.find(i)+1) * len(a)**index
        return number



    def GetLastRow(self, spreadsheet_id, sheet_name, sheet_id):
        '''Метод возвращает номер последней заполненной строки на листе '''
        try:
            res = self.AppendNewRows(spreadsheet_id, sheet_name,"A1",[["*****",
"!!!"]] ,"INSERT_ROWS" )
            index_last_row = res['tableRange']
            index_last_row = index_last_row.split(':')
            index_last_row = re.findall(r'\d+',index_last_row[-1])[-1]
            self.DeleteRows(spreadsheet_id, sheet_id,
str(int(index_last_row)),str(int(index_last_row)+1))
            return index_last_row
        except KeyError as e:
            logger.error('На листе нет заполненных строк, возвращено значение 1')
            return 1
```

```
        except Exception as e:
            logger.error('При попытке нахождения последней строки возникла ошибка:
{0}, {1}'.format(type(e), e))

    def GetAllSheetData(self, spreadsheet_id, sheet_name, sheet_id, first_sell =
'A1', last_sell = 'ZZ10000'):
        major_dimension = "ROWS"
        res = self.service.spreadsheets().values().get(spreadsheetId =
spreadsheet_id, range = '{0}!{1}:{2}'.format(sheet_name,first_sell,last_sell),
majorDimension = major_dimension).execute()
        return res['values']
```

Приложение Г – Акт о внедрении

ИНН/КПП 2466268568/246601001
ОГРН 1132468067214
ОКПО 21940204
660017, г. КРАСНОЯРСК,
УЛ. КИРОВА, Д. 19, ОФ. 49
ТЕЛ. (391) 288-02-89,
ФАКС (391) 212-04-72

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСВЕННОСТЬЮ
СПЕЦПРОМАВТОМАТИКА

Акт о внедрении

результатов бакалаврской работы Мартынова В.Е.

«Разработка информационно-аналитической системы формирования отчетности на предприятии (на примере ООО «СпецПромАвтоматика»)

Настоящий акт свидетельствует, что информационно-аналитическая система, описанная в бакалаврской работе специалиста по сопровождению ПО, Мартынова В.Е., внедрена в бизнес-процессы ООО «СпецПромАвтоматика». По результатам разработки и внедрения получены положительные результаты:

— повышена эффективность отдела информационной аналитики;
— разработан алгоритм автоматического формирования отчета по подрядчикам;
— разработан алгоритм автоматического формирования отчета по колесной и тяжелой техники холдинга «Сибзолото»;
— разработан алгоритм автоматического формирования отчета по заправкам техники АТЗ;
— внедрено использование файлового хостинга Google Docs для решения производственных задач.

Генеральный директор **Ф.В. Мараескул**

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт управления бизнес-процессами и экономики
Кафедра «Бизнес-информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.Н. Пупков
«_1_» _07_ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.03.02 «Прикладная информатика в менеджменте»

Разработка информационно-аналитической системы формирования отчетности на предприятии (на примере ООО «СпецПромАвтоматика»)

Руководитель _____ Доцент кафедры «БИ» А.В. Чубаров
подпись, дата

Консультант _____ Доцент кафедры «ЭУБП» Г.Ф. Яричина
подпись, дата

Выпускник _____ В.Е. Мартынов
подпись, дата

Нормоконтролер _____ Д.В.Спиридонов
подпись, дата

Красноярск 2019