Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
институт
Информатика
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Рубан
«___»_____ 2016 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Модернизация клиентской части информационной системы спутникового мониторинга на транспорт

231000.62 «Программная инженерия»

Научный руководитель/
руководитель _____ доцент, кандидат тех. наук А.С. Кузнецов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Н.Е. Пятов
подпись дата инициалы, фамилия

Нормоконтролер _____ О.А. Антамошкин
подпись дата инициалы, фамилия

Красноярск 2016

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
институт
Информатика
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Рубан
«_20_» ___12___ 2016 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Пятову Никите Евгеньевичу
фамилия, имя, отчество

Группа КИ12-18Б Направление (специальность) 231000.62
номер код

Программная инженерия
наименование

Тема выпускной квалификационной работыМодернизация клиентской части информационной системы спутникового мониторинга на транспорт

Утверждена приказом по университету № 6145/с от 10.05.2016

Руководитель ВКР А.С. Кузнецов доцент, кандидат тех. наук
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР клиентская часть информационной системы спутникового мониторинга на транспорт “Импульс АРМ”

Перечень разделов ВКР Анализ предметной области, Постановка задачи, Выбор технологий, Реализация

Перечень графического материала архитектурасистемы спутникового мониторинга транспорта, графический редактор шаблона отчета, пример структуры *xml файла шаблона отчета, диаграмма классов, доступ к созданию отчета в главном диалоговом окне, доступ к созданию отчетов через выпадающее меню, интерфейс создания общего отчета, Просмотр отчета в формате PDF

Руководитель ВКР _____ А.С. Кузнецов
подпись инициалы и фамилия

Задание принял к исполнению _____ И.Е. Пятов
подпись, инициалы и фамилия студента

« ___ » _____ 20__ г.

АННОТАЦИЯ

Выпускная квалификационная работа по теме “Модернизация клиентской части информационной системы спутникового мониторинга на транспорт” содержит 35 страницы текстового документа, 12 использованных источников, 10 иллюстраций, 2 приложения.

В ходе выполнения данной работы был рассмотрен принцип работы систем спутникового мониторинга транспорта. Проанализированы Java-библиотеки для генерации отчетов и выбран наиболее подходящий. Был реализован дополнительный функционал построения и формирования отчетов для системы “Ипульс АРМ”.

ANNOTATION

Final qualifying work on the theme "Modernization of the client part of the information system of satellite monitoring of transport" contains 35 pages of a text document, 12 sources used, 10 illustrations, 2 applications.

In the course of this work it was considered the principle of operation of the satellite monitoring of transport systems. Analyzes Java-library for generating reports and selected the most suitable. build additional functionality and reporting it has been implemented for the "Pulses ARM" system.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....7
ОСНОВНАЯ ЧАСТЬ.....9
1. Анализ предметной области.....9
2. Постановка задачи.....12
2.1. Цели и задачи.....12
2.2. Требования к интерфейсу.....12
2.3. Требования к отчетам.....13
3. Выбор технологий.....14
3.1. Инструмент построения отчетов.....14
4. Реализация.....16
4.1. Архитектура классов.....16
4.2. Графический интерфейс создания отчетов.....18
ЗАКЛЮЧЕНИЕ.....24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....25
Приложение А Исходный код некоторых функций.....26

ВВЕДЕНИЕ

В настоящее время потребности социально-экономической сферы любой страны не обходятся без транспортной системы. Транспорт, наряду с другими инфраструктурными отраслями, обеспечивает базовые условия жизнедеятельности общества. Он связывает между собой различные районы, предприятия и отрасли народного хозяйства, играет огромную роль при размещении предприятий, заводов и является важным условием рационального расположения производства. Особенное значение транспорт имеет для России, так как именно на такой обширной территории для единства и целостности государства необходима развитая транспортная система.

С появлением и развитием спутниковых систем навигации и технологий сотовой связи широкое распространение получили системы спутникового мониторинга транспорта. Сейчас они позволяют в режиме реального времени предотвращать несанкционированные действия водителей, обеспечивать сохранность перевозимого груза, контролировать расхода топлива и т.д. Что несомненно сказывается на увеличение эффективности работы транспортных предприятий и всю транспортную систему в целом.

В свою очередь, большая редкость когда компании выпускают готовое программное обеспечение в законченной версии. Программный продукт может устареть, может появится необходимость усовершенствовать программу под текущие требования или новые платформы. Компаниям нужно оставаться конкурентоспособными и делать свои программные продукты более удобными и современными. Поэтому практически каждое программное обеспечение не обходится без обновления и модернизации, чтобы максимально удовлетворить потребности пользователя.

Информационная система спутникового мониторинга “Импульс АРМ” компании “ИМПУЛЬС-ГЛОНАСС” не является исключением. Причиной модернизации для этой информационной системы стала необходимость добавления функционала связанного с обработкой и наглядным

представлением данных. Было решено добавить формирование и построение отчетов в формате PDF.

ОСНОВНАЯ ЧАСТЬ

1. Анализ предметной области

Система спутникового мониторинга транспорта - система, построенная на основе систем 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 и offline - информация считывается по прибытию на диспетчерский пункт.

На транспортном средстве 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.

Мобильный модуль может быть построен на основе приёмников спутникового сигнала, работающих в стандартах NAVSTAR GPS или ГЛОНАСС. 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. При этом, в сравнении с NAVSTAR GPS, система ГЛОНАСС пока работает менее надёжно и в совокупности с наземным оборудованием даёт бóльшую погрешность

вычисления местоположения абонента. Клиентское оборудование ГЛОНАСС стоит дороже, имеет бóльшие размеры и худшие параметры энергопотребления, представлено на рынке не так широко, как GPS. Этим объясняется сложность внедрения ГЛОНАСС-мониторинга и вынужденное его использование государственными предприятиями России.

Рисунок 1 - архитектура системы спутникового мониторинга транспорта

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.

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)

Клиентская часть системы “Импульс АРМ” реализована на языке программирования Java с ипользованием NetBeans Platform.

NetBeans Platform – это обширная платформа Java, на которой могут базироваться крупномасштабные приложения для рабочей среды. Платформа NetBeans содержит интерфейсы API, которые упрощают обработку окон, действий, файлов и многих других объектов, а также ситуаций, типичных для приложений.

Каждая возможность в приложении на платформе NetBeans может обеспечиваться отдельным модулем NetBeans, который сравним с подключаемым модулем. Модуль NetBeans – это группа классов Java, которая предоставляет приложение с определенными возможностями.

2. Постановка задачи

2.1. Цели и задачи

Цель работы: добавление функционала в клиентскую часть системы “Импульс АРМ” позволяющего формирование и построение отчетов.

Задачи работы:

1. Создать графический интерфейс построения отчетов
2. Найти подходящий инструмент для построения отчетов

2.2. Требования к интерфейсу

Необходимо создать графический интерфейс создания для следующих видов отчетов:

- Отчет по датчикам
- Отчет по событиям

- Общий отчет
- Отчет по выгрузке бункера
- Свободный отчет
- Покрытие сотовой связи
- Регистрация в системе

Интерфейс должен соответствовать общему стилю системы и быть интуитивно понятным. В случае долгого построения отчета иметь какой-либо индикатор или уведомления процесса работы. Если произошла ошибка формирования и построения отчета так же уведомлять об этом пользователя.

Все окна создания отчетов должны содержать: компонент выбора транспорта по которому делается отчет с возможностью множественного выбора, компонент выбора периода времени за который требуется сделать отчет, группы параметров с специфическим набором для каждого отчета. Группы компонентов отвечающих за настройку параметров каждого отчета должны быть правильно сгруппированы и удобно расположены.

Должны быть предусмотрены все некорректные ситуации выбора параметров создания отчета. Уведомлять пользователя в случае попытки создания некорректного отчета. В случаи если возможно объяснение проблемы, то пояснить пользователю в чем заключается проблема.

Для просмотра отчета необходимо отдельное окно.

2.3. Требования к отчетам

Отчеты должны иметь одинаковую структуру. Каждый отчет должен содержать:

- Наименование отчета
- ФИО дистпетчера
- Дата составления
- Период
- Таблица

В таблице обязательно должно присутствовать поле с наименованием собственника транспорта по которому делался отчет. Остальные поля таблицы варьируются в зависимости от вида отчета.

Отображение отчета должно быть корректным и в формате PDF. В случаи, если таблица занимает больше одной страницы, то перенос таблицы так же должен быть корректно отображен.

3. Выбор технологий

3.1. Инструмент построения отчетов

В первую очередь отбирались библиотеки для языка программирования Java, бесплатные и поддерживающие генерацию документов в формат PDF.

Были выбраны следующие библиотеки:

- JasperReports
- iText
- DynamicReports

В дальнейшем представленные библиотеки оценивались по следующим критериям:

- удобство
- популярность и развитие сообщества

Удобство рассматривалось в первую очередь с стороны скорости разработки отчета. То есть быстрого составления нужного шаблона, чтобы при этом компоненты данной библиотеки полностью удовлетворяли поставленной задаче.

Критерий популярности и развития сообщества тоже важен, так как от этого зависит скорость ответа на заданные вопросы в сообществе при его возникновении или есть большая вероятность найти ответ в уже имеющихся на форуме.

В итоге была выбрана библиотека JasperReports, потому что она оказалась самой популярной библиотекой из представленных, а также имеется интеграция(подключаемый плагин) с средой разработки, который представляет графический редактор на рис. 2. Этот редактор позволяет не составлять xml файл самому, а составляет его сам. Нужно только правильно расположить компоненты на шаблоне и задать правильные свойства.

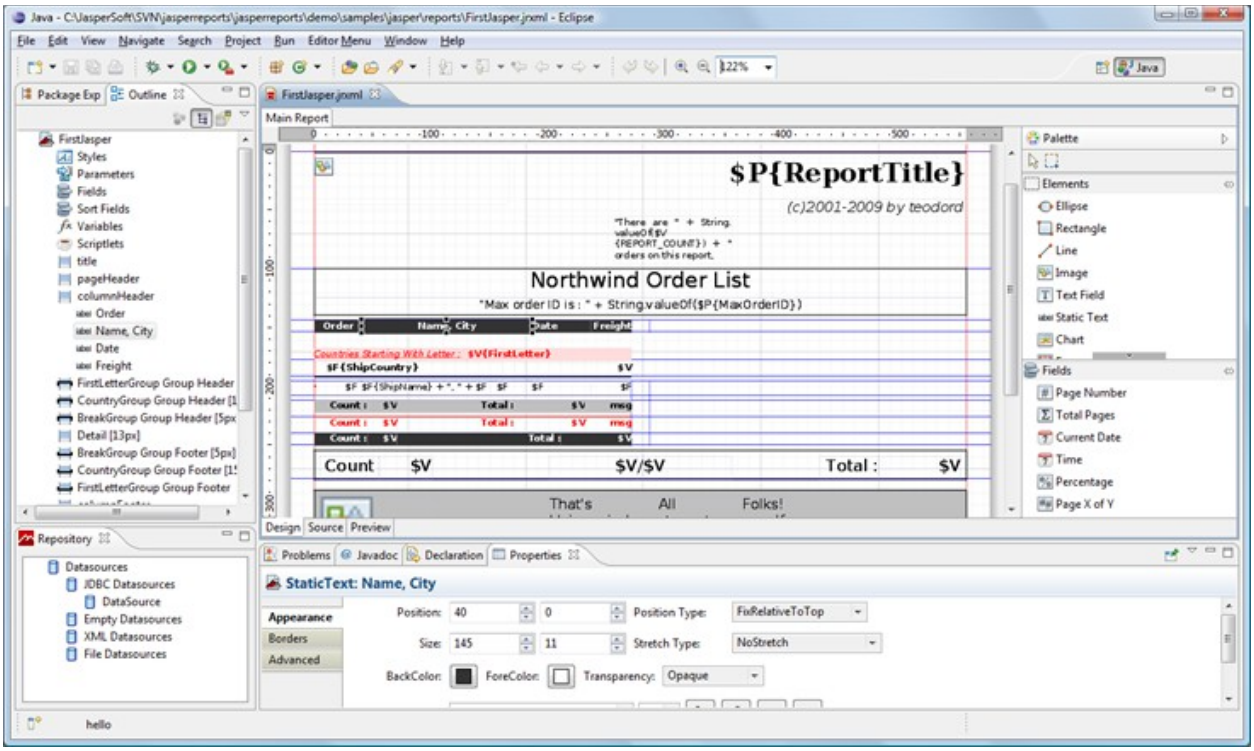

Рисунок 2 - Графический редактор шаблона отчета

DynamicReports представляла только возможность создавать шаблон программно. Нужно составлять xml файл вручную и при этом соблюдать правильную структуру, которая отличается от какой-то общей принятой. Если допустить малейшую опечатку в структуре файла, то возникшую ошибку сложно найти. Сообщение об ошибке малоинформативно, что приводит к потере неопределенного количества времени. К тому же это добавляет некоторые неудобства, так как нужно запоминать новые элементы и обращаться лишний раз к документации. Пример структуры шаблона на рис. 3.

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE jasperReport
  PUBLIC "-//JasperReports//DTD Report Design//EN"
  "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">

<jasperReport name="ReportName">

    <style name="Arial_Normal" isDefault="true" fontName="Arial"
        fontSize="12" pdfFontName="c:\tahoma.ttf" pdfEncoding="Cp1251"
        isPdfEmbedded="false" />

    <field name="name" class="java.lang.String" />
   <detail>
        <band height="20">
            <textField>
                <reportElement x="0" y="0" width="50" height="20" />
                <textFieldExpression class="java.lang.String">
                    <![CDATA[$F{name}]]>
                </textFieldExpression>
            </textField>
        </band>
    </detail>
</jasperReport>
```

Рисунок 3 - Пример структуры *xml файла шаблона отчета

iText библиотека имеет свой редактор, но он не имел интеграции с средой разработки. Редактор был отдельной программой. Компоненты iText нуждались в большей настройке свойств, поэтому являлась менее мощной библиотекой.

4. Реализация

4.1. Архитектура классов

Классы:

- Report - Класс обрабатывающий основные события при создании отчета
- ReportOptionsPanel - Класс компонента дополнительных параетров для отчета
- BuildReportDialogLauncher - Класс реализующий сервис, который подгружает нужные виды транспорта под разны виды отчетов
- ReportPanel - Суперкласс содержащий общие компоненты для всех отчетов

- GroupingPanel - Класс компонента включающего группировку элементов в таблице
- ReportView - Класс создающий окно просмотра сгенерированного отчета в PDF
- SourceReportData - Класс хранящий все данные передаваемые с сервера и передающий их в шаблон отчета
- EventOptionPanel - Класс наследуюемый от ReportPanel предназначен для отчета по событиям
- SummaryPanel - Класс наследуюемый от ReportPanel предназначен для общего отчета
- ReportEventAction - Класс отображающий компонент выбора отчета по событиям в меню и обрабатывающий его выбор
- SummaryReportAction - Класс отображающий компонент выбора общего отчета в меню и обрабатывающий его выбор
- SensorsReportAction - Класс отображающий компонент выбора отчета по датчикам в меню и обрабатывающий его выбор
- SensorsIntervalReportAction - Класс отображающий компонент отчета по интервалам датчиков в меню и обрабатывающий его выбор
- RegistrationReport - Класс отображающий компонент отчета по регистрации в сети в меню и обрабатывающий его выбор

На рис. 4 отображена диаграмма основных классов. ReportPanel является суперклассом который хранит общие элементы всех отчетов, а наследующие его классы уже добавляют соответствующие им параметры.

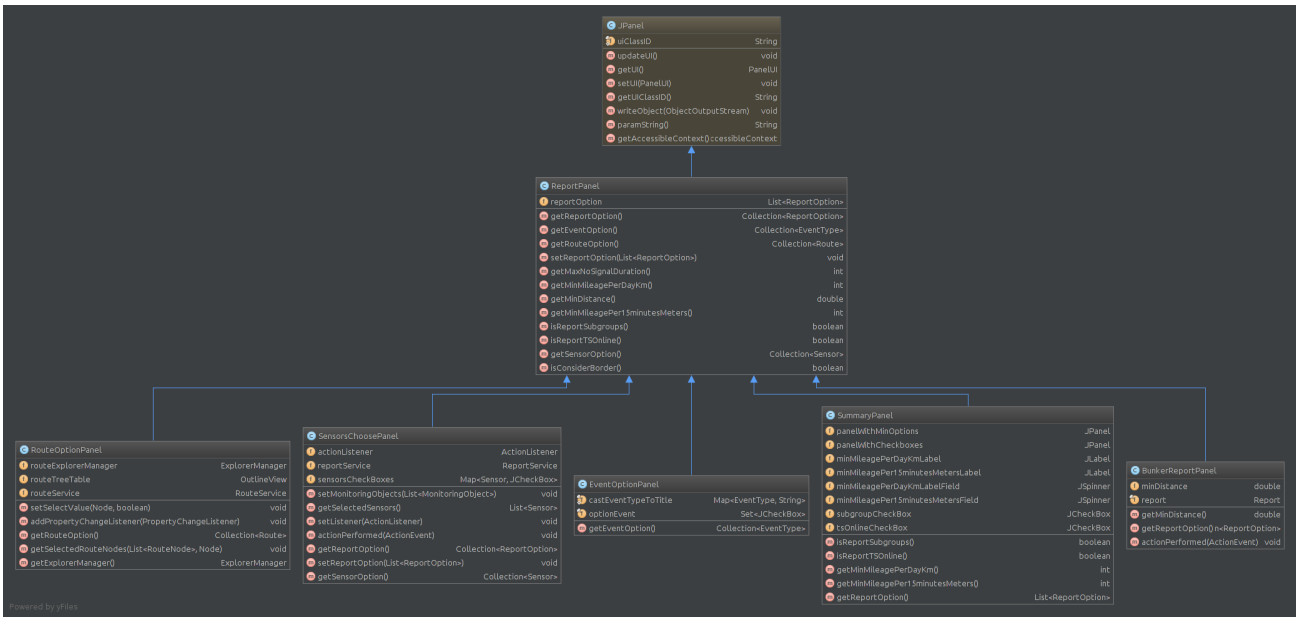

Рисунок 4 - Диаграмма классов

4.2. Графический интерфейс создания отчетов

Для быстрого перехода к нужному отчету были созданы кнопки в меню(рис. 5). В эту часть интерфейса были вынесены отчеты, которые были приняты более важными и часто составлямыми. Им сделаны иконки.

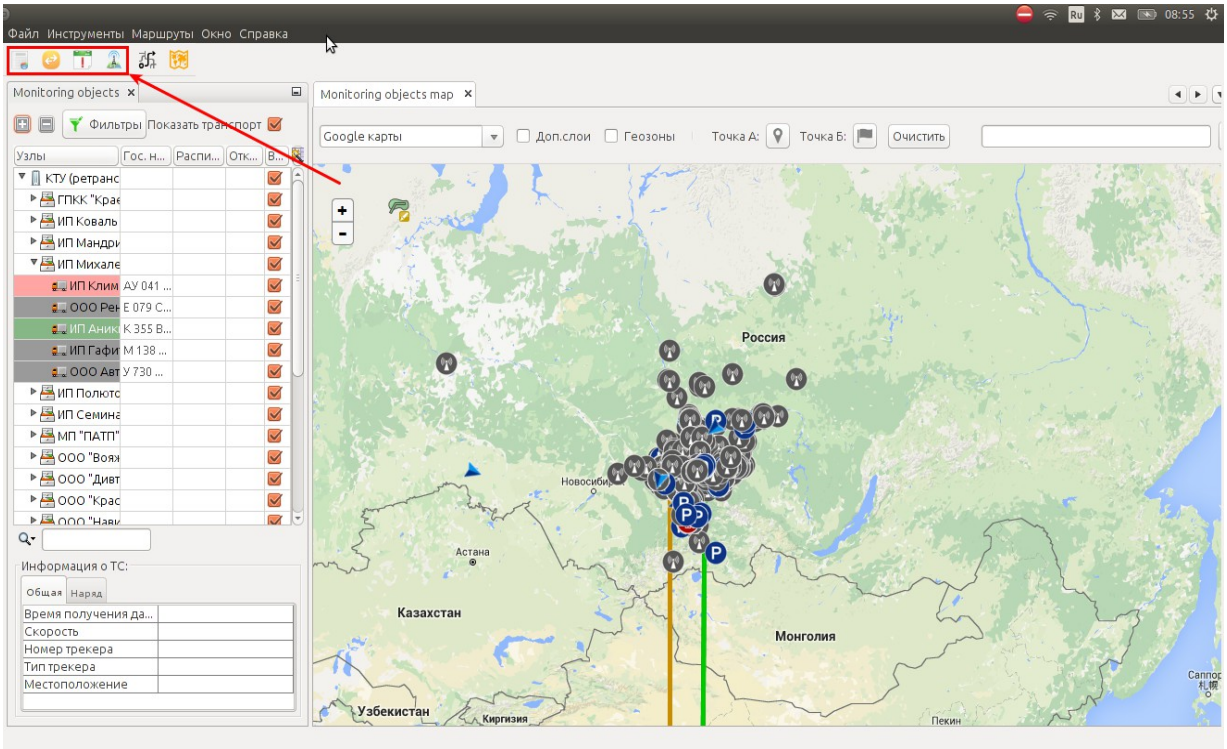

Рисунок 5 - Доступ к созданию отчета в главном диалоговом окне

Переход к созданию отчета также возможен через выпадающее меню в вкладке инструменты(рис. 6). Здесь можно просмотреть полный список отчетов и заметить сочетание горячих клавиш.

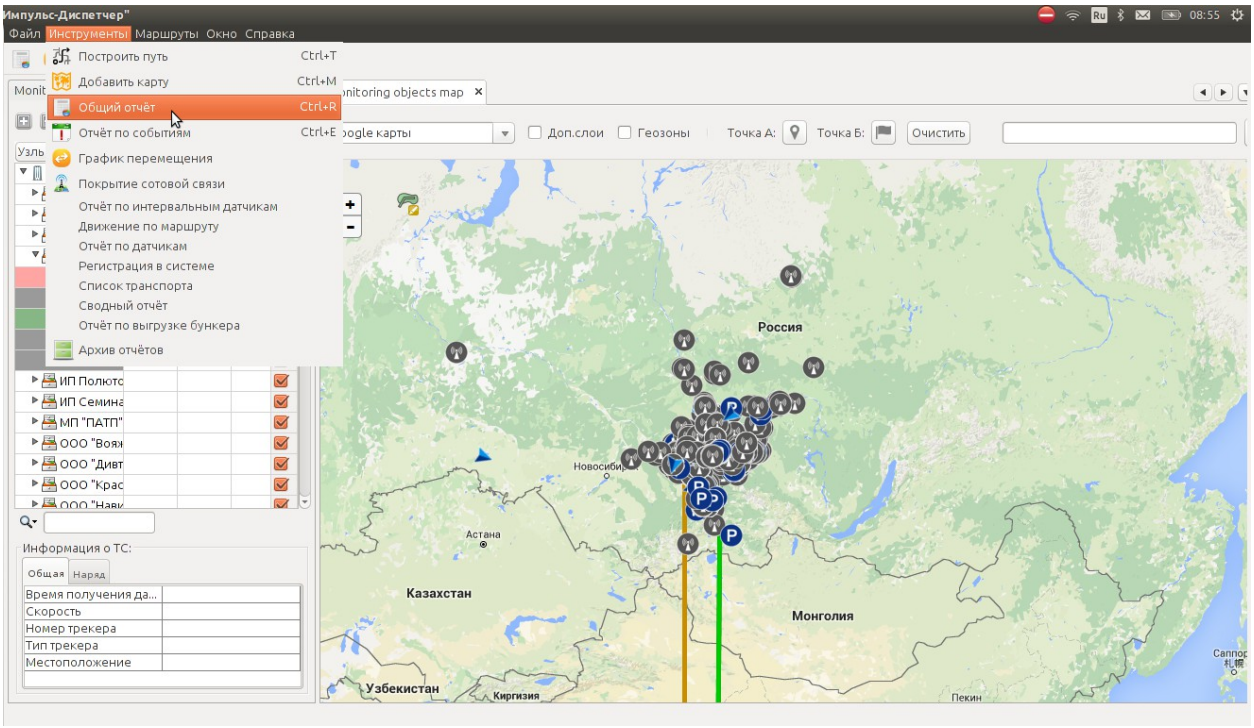

Рисунок 6 - Доступ к созданию отчетов через выпадающее меню

При выборе общего отчета появляется соответствующие диалоговое окно(рис.7). На окне “Построитель отчета” можно зрительно заметить три блока элементов:

- Дерево транспорта
- Дата
- Параметры

Дерево транспорта имеет трехуровневую структуру:

- Первый уровень - Тип транспорта
- Второй уровень - Копании
- Третий уровень - Транспорт

Дата выбирается за определенный период времени. Над элементами выбора даты показываются наиболее часто употребляемые временные отрезки.

Параметры делятся еще на четыре группы:

- Скорость
- Пробег

- Топливо
- Моточасы

По умолчанию параметры с чекбоксами выбраны.

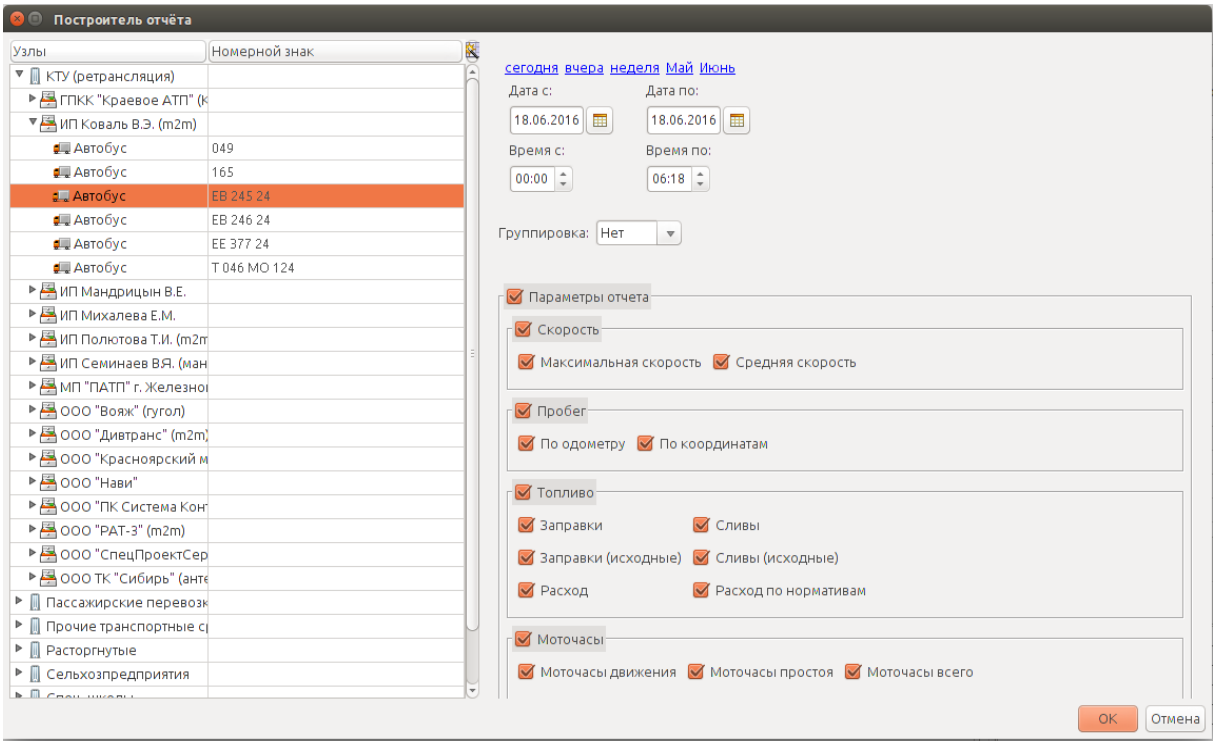

Рисунок 7 - Интерфейс создания общего отчета

Как уже говорилось. Блок с параметрами данного построителя отчета изменился в зависимости от вида отчета(рис. 6).

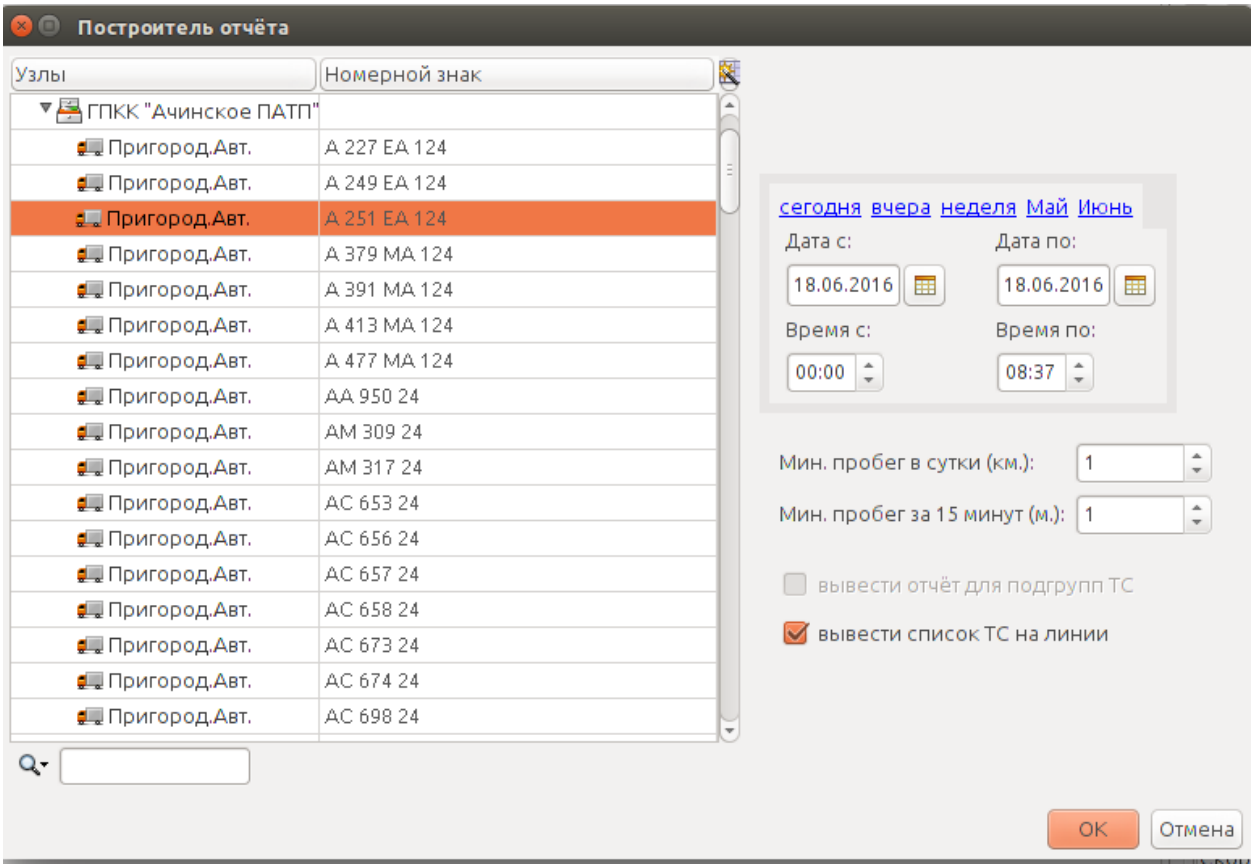

Рисунок 8 - Интерфейс создания сводного отчета

Для создания отчета по события требуются параметры с типами событий(рис. 7).

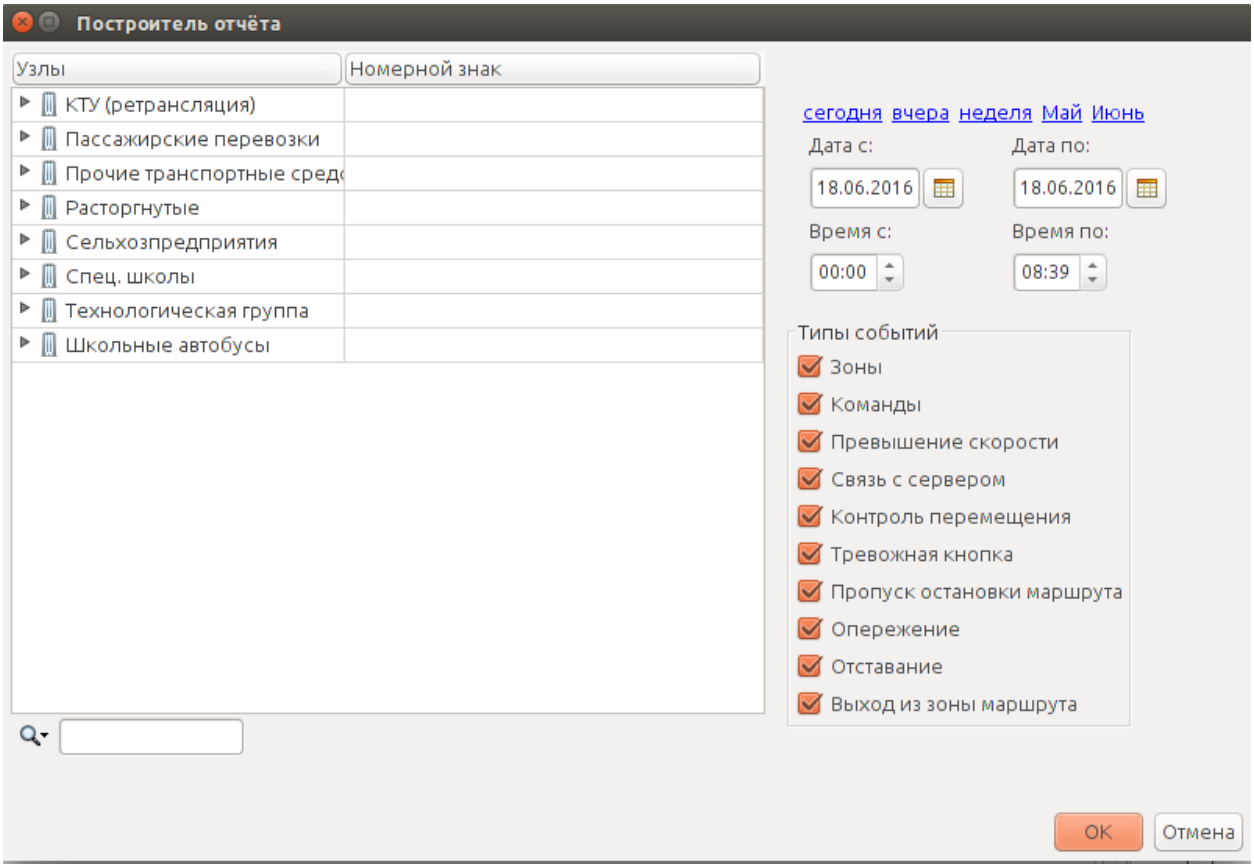

Рисунок 9 - Интерфейс создания отчета по событиям

После создания отчета он открывается в отдельной вкладке программы в формате PDF. В этой вкладке можно просмотреть все страницы отчета, изменить масштаб, а также сохранить или распечатать документ.

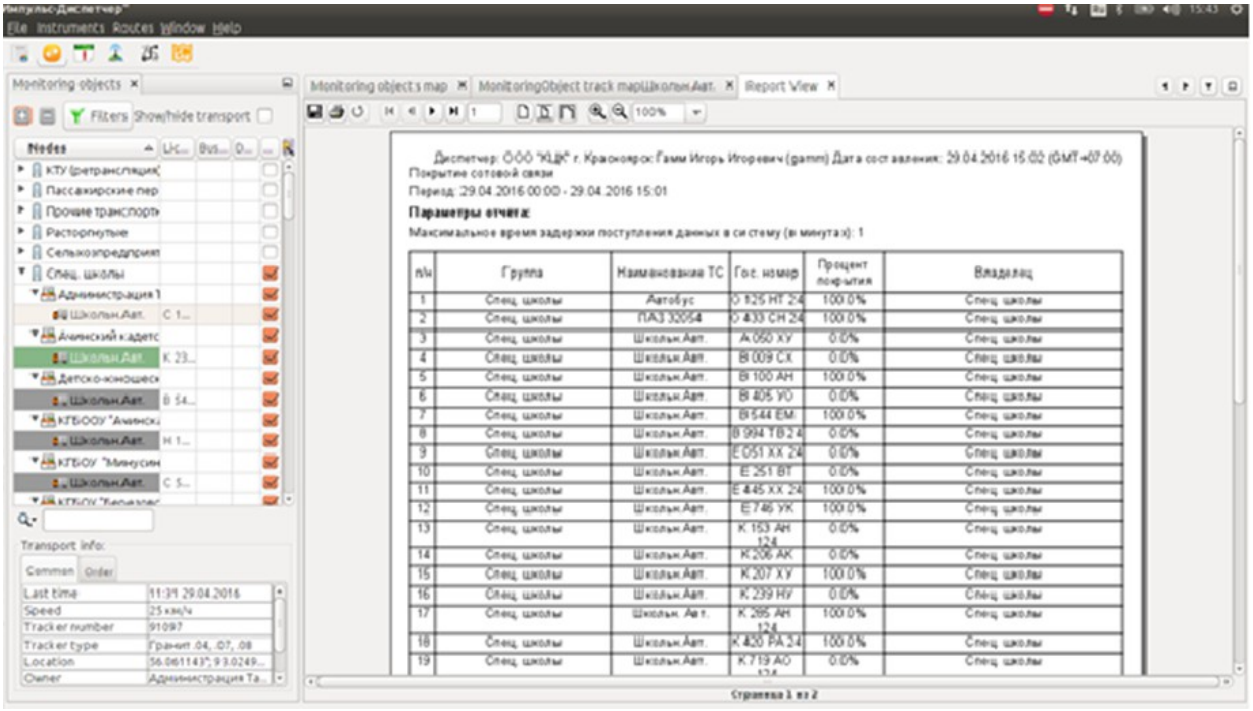

Рисунок 10 - Просмотр отчета в формате PDF

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе была рассмотрена информационная система спутникового мониторинга транспорта “Импульс АРМ”. Проведен анализ существующих Java-библиотек для построения отчетов и выбран наилучший подходящий вариант. Изучен принцип работы подобных систем и проведена модернизация с учетом требований к интерфейсу и построению отчетов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://itextpdf.com/
2. https://netbeans.org/features/platform/
3. http://community.jaspersoft.com/project/jasperreports-library
4. http://www.dynamicreports.org/
5. http://cyberleninka.ru/article/n/problemy-sootvetstviya-transportnoy-sistemy-potrebnostyam-sotsialno-ekonomicheskogo-razvitiya-strany
6. https://creativeconomy.ru/articles/30993/
7. http://teoria-practica.ru/rus/files/arhiv_zhurnala/2014/4/ekonomika/khegay.pdf
8. http://www.tehprog.ru/index.php_page=lecture0141.html
9. https://rg.ru/2008/09/03/glonass-dok.html
10. http://www.interface.ru/home.asp?artId=34854
11. http://ru.intechcore.com/modernizacija-programmnogo-obespechenija/
12. http://gps-club.ru/gps_test/detail.php?ID=11006

Приложение А Исходный код некоторых функций

Report.java

```
@NbBundle.Messages({
    "CTL_ReportDialogTitle=Report",
    "CTL_ReportDialogNoMonitoringObjectsSelectedError=Error",
    "CTL_ReportDialogNoMonitoringObjectsSelectedErrorMessage=You should select one ore more monitoring objects.",
    "CTL_LoadingReport=Loading reports",
    "CTL_ReportDialogNoEventOptionsSelectedErrorMessage=You should select one ore more Event options.",
    "CTL_ReportDialogNoRouteOptionsSelectedErrorMessage=You should select one ore more route options.",
    "CTL_ReportDialogNoGeneralOptionsSelectedErrorMessage=You should select one ore more parameters.",
    "CTL_NoDataMessage=No data",
    "MSG_YouHaventRight=You haven't right for this  action."
})
public class Report implements ActionListener, ReportDoneListener {
  private static final Logger LOGGER = Logger.getLogger(Report.class.getName());
  private final ReportParameter.ReportType reportType;
  ReportParamsPanel reportParamsPanel;
  private ReportPanel reportPanel;
  final List<MonitoringObject> monitoringObjects = new ArrayList<>();
```

```
  private Map<Set<ReportParameter.ReportOption>, Set<TrackOption>>
reportOptionRelationTrackOptions = new HashMap(){{

    put(Sets.newHashSet(ReportParameter.ReportOption.AVG_SPEED,
ReportParameter.ReportOption.MAX_SPEED),

        Sets.newHashSet(TrackOption.SEGMENT_NA_MOOVEMENT,
TrackOption.SEGMENT_NA_NAVIGATION/*,
TrackOption.INTERVALS_PARKING,

        TrackOption.INTERVALS_MOVEMENT,
TrackOption.INTERVALS_IGNITION_ONLINE,
TrackOption.INTERVALS_MOTOHOURS_PARKING,

        TrackOption.INTERVALS_MOTOHOURS_MOVEMENT,
TrackOption.SEGMENT_SA_MACHINEHOUR,
TrackOption.NAVIGATIONS_PARKING,

        TrackOption.NAVIGATIONS_MOVEMENT*/));

    put(Sets.newHashSet(ReportParameter.ReportOption.DRAINING,
ReportParameter.ReportOption.DRAININGSOURCE,

        ReportParameter.ReportOption.FUELING,
ReportParameter.ReportOption.FUELINGSOURCE),

        Sets.newHashSet(TrackOption.FUEL_DRAINING,
TrackOption.SEGMENT_SA_FUEL, TrackOption.FUEL_AGGREGATES));

    put(Sets.newHashSet(ReportParameter.ReportOption.CONSUMPTION,
ReportParameter.ReportOption.CONSUMPTION_NORM),

        Sets.newHashSet(TrackOption.SEGMENT_NA_MOOVEMENT,
TrackOption.SEGMENT_NA_NAVIGATION/*,
TrackOption.SEGMENT_SA_EQUIPMENT,

        TrackOption.INTERVALS_PARKING,
TrackOption.INTERVALS_MOVEMENT,
TrackOption.INTERVALS_IGNITION_ONLINE,

        TrackOption.INTERVALS_MOTOHOURS_PARKING,
TrackOption.INTERVALS_MOTOHOURS_MOVEMENT,
TrackOption.SEGMENT_SA_MACHINEHOUR,

        TrackOption.NAVIGATIONS_PARKING,
TrackOption.NAVIGATIONS_MOVEMENT*/));
```

```
put(Sets.newHashSet(ReportParameter.ReportOption.MOTOHOURS_IDLE,
ReportParameter.ReportOption.MOTOHOURS_MOVEMENT,
ReportParameter.ReportOption.MOTOHOURS_SUMMARY),

        Sets.newHashSet(TrackOption.SENSOR_IGNITION,
TrackOption.SENSOR_EQUIPMENT,
TrackOption.SEGMENT_SA_EQUIPMENT));

    put(Sets.newHashSet(ReportParameter.ReportOption.MILEAGE,
ReportParameter.ReportOption.MILEAGE_OVER_COORDINATES),

        Sets.newHashSet(TrackOption.SEGMENT_NA_MOOVEMENT,
TrackOption.SEGMENT_NA_NAVIGATION));

  }};

  public Report(ReportParameter.ReportType reportType, ReportPanel
reportPanel) {

    this.reportType = reportType;

    this.reportPanel = reportPanel;

  }

  @Override

  public void actionPerformed(ActionEvent e) {

    final DialogDescriptor reportsDialogDescriptor = new DialogDescriptor(

        reportParamsPanel, Bundle.CTL_ReportDialogTitle(), true, null

    );

    final ReportDoneListener reportDoneListener = this;

    reportsDialogDescriptor.setButtonListener(

        new ActionListener() {

          @Override

          public void actionPerformed(ActionEvent actionEvent) {

            if (actionEvent.getSource() ==
DialogDescriptor.CANCEL_OPTION) {
```

```
                reportsDialogDescriptor.setClosingOptions(null);
            }
            else if (actionEvent.getSource() ==
DialogDescriptor.OK_OPTION && isValidOptions(reportParamsPanel)) {
                reportsDialogDescriptor.setClosingOptions(null);
                //Запуск отчета
                Set<TrackOption> trackOptions =
prepareTrackOptions(reportType, reportParamsPanel,
reportOptionRelationTrackOptions);
                SourceReportData sourceReportData = new
SourceReportData(
                        monitoringObjects,
                        reportParamsPanel.getReportGroup(),
                        reportParamsPanel.getReportPanel().getReportOption(),
                        reportParamsPanel.getReportPanel().getEventOption(),
                        reportParamsPanel.getReportPanel().getRouteOption(),
                        reportParamsPanel.getReportPanel().getSensorOption(),
                        reportPanel.getMaxNoSignalDuration(),
                        reportPanel.getMinMileagePerDayKm(),
                        reportPanel.getMinMileagePer15minutesMeters(),
                        reportPanel.getMinDistance(),
                        reportPanel.isReportSubgroups(),
                        reportPanel.isReportTSOnline(),
                        reportParamsPanel.onlyTable(),
                        reportPanel.isConsiderBorder()
                );
                ReportGetService reportGetService = new
ReportGetService(reportDoneListener);
```

```
reportGetService.getReport(reportParamsPanel.getStartDate(),
reportParamsPanel.getEndDate(), reportType,
                        trackOptions, sourceReportData);
                }
            }
        }
    );
    reportsDialogDescriptor.setClosingOptions(new Object[]{});
     DialogDisplayer.getDefault().notifyLater(reportsDialogDescriptor);
 }
 private Set<TrackOption>
prepareTrackOptions(ReportParameter.ReportType reportType,
ReportParamsPanel reportParamsPanel,
                                    Map<Set<ReportParameter.ReportOption>,
Set<TrackOption>> reportOptionRelationTrackOptions) {
    Set<TrackOption> trackOptions = TrackOption.ALL;
    if (reportType == ReportParameter.ReportType.GENERAL) {

reportParamsPanel.getReportPanel().setReportOption(reportParamsPanel.get
ReportOptionsPanel().getSelectedParameters());
        for (ReportParameter.ReportOption ro:
reportParamsPanel.getReportOptionsPanel().getSelectedParameters()) {
            for (Set<ReportParameter.ReportOption> roSet:
reportOptionRelationTrackOptions.keySet()) {
                if (roSet.contains(ro))
trackOptions.addAll(reportOptionRelationTrackOptions.get(roSet));
            }
        }
    }
```

```
    if (reportType == ReportParameter.ReportType.MOVE) {
        trackOptions =
Sets.newHashSet(TrackOption.NAVIGATIONS_MOVEMENT,
TrackOption.NAVIGATIONS_PARKING, TrackOption.INTERVALS,
            TrackOption.SEGMENT_NA_NAVIGATION,
TrackOption.SEGMENT_NA_MOOVEMENT);
    }
    return trackOptions;
  }

  private boolean isValidOptions(ReportParamsPanel reportParamsPanel) {
    if (reportParamsPanel.getMonitoringObjects().isEmpty()) {

showOptionsErrorMessage(Bundle.CTL_ReportDialogNoMonitoringObjects
SelectedErrorMessage());
        return false;
    }
    if (reportType == ReportParameter.ReportType.GENERAL
        && !isValidReportOptions()
        ) {

showOptionsErrorMessage(Bundle.CTL_ReportDialogNoGeneralOptionsSel
ectedErrorMessage());
        return false;
    }
    if (reportType == ReportParameter.ReportType.EVENT
        && !isValidEventOptions()
        ) {
```

```
showOptionsErrorMessage(Bundle.CTL_ReportDialogNoEventOptionsSelect
edErrorMessage());
        return false;
    }
    if (reportType == ReportParameter.ReportType.ROUTE
        && !isValidRouteOptions()
        ) {

showOptionsErrorMessage(Bundle.CTL_ReportDialogNoRouteOptionsSelect
edErrorMessage());
        return false;
    }
    monitoringObjects.clear();
    monitoringObjects.addAll(reportParamsPanel.getMonitoringObjects());
    return true;
  }

  private void showOptionsErrorMessage(String errorMessage) {
    JOptionPane.showMessageDialog(
        WindowManager.getDefault().getMainWindow(),
        errorMessage,
        Bundle.CTL_ReportDialogNoMonitoringObjectsSelectedError(),
        JOptionPane.ERROR_MESSAGE
    );
  }
  private boolean isValidEventOptions() {
    return !reportPanel.getEventOption().isEmpty();
  }
```

```
  private boolean isValidRouteOptions() {
     return !reportPanel.getRouteOption().isEmpty();
  }
  private boolean isValidReportOptions() {
     return !
reportParamsPanel.getReportOptionsPanel().getSelectedParameters().isEmpt
y();
  }
  public void updateReportPanel() {
     reportParamsPanel = new ReportParamsPanel(
           reportPanel,
           reportType,
           reportType == ReportParameter.ReportType.TRANSPORTS_LIST
                || reportType ==
ReportParameter.ReportType.MONITORING_FOR_SCHOOL?
                true:
                false,
           reportType == ReportParameter.ReportType.TRANSPORTS_LIST
                || reportType ==
ReportParameter.ReportType.REGISTRATION?
                true:
                false
     );
  }
  @Override
  public void receiveReport(byte[] reportBinary) {
     if (reportBinary == null)
        return;
```

```
    try {
        final JasperPrint finalJasperPrint = (JasperPrint) new
ObjectInputStream(new ByteArrayInputStream(reportBinary))
                .readObject();
        if (finalJasperPrint.getPages().size() > 0) {
            SwingUtilities.invokeLater(
                    new Runnable() {
                        @Override
                        public void run() {
                            final TopComponent reportView = new
ReportView(finalJasperPrint);
                            Mode modeDocking =
WindowManager.getDefault().findMode("editor");
                            if (modeDocking != null) {
                                modeDocking.dockInto(reportView);
                            }
                            reportView.open();
                            reportView.requestActive();
                        }
                    }
            );
        } else {
            DialogDisplayer.getDefault().notify(
                    new DialogDescriptor.Message(Bundle.CTL_NoDataMessage(),
                            DialogDescriptor.ERROR_MESSAGE)
            );
        }
    } catch (ClassNotFoundException e) {
```

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