УТВЕРЖДЕН

НПБК.20765-01 13 01-ЛУ

ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА

Описание программы

НПБК.20765-01 13 01

Листов 25

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

2022

Литера O_1

АННОТАЦИЯ

Настоящий документ представляет собой описание програмы «Технологическая платформа» НПБК.20765-01 и содержит общие сведения о программе, описание функционального назначения и логической структуры. В документе приводится перчень используемых технических средств, а также описание входных и выходных данных.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Наименование программы: «Технологическая платформа».

Языки программирования: C++ и Java.

Обозначение: НПБК.20765-01.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Назначение программы

Назначение «Технологической платформы» НПБК.20765-01 заключается в обеспечении возможности формирования отчетных документов по указанным шаблону и параметрам. Выполнение требования назначения может осуществляться двумя возможными способами:

1) через запуск программы с использованием командной строки ОС;

2) через включение в состав других ПИ, функционирующих в среде «Технологической платформы», и вызов с помощью программных интерфейсов для приложений, функционирующих в среде «Технологической платформы».

2.2. Классификация решаемых задач

«Технологическая платформа» предназначена для формирования отчетных документов по указанным шаблону и параметрам. В состав параметров входит:

- xml-файл, описывающий правила формирования отчета;
- формат отчетного документа;
- путь для сохранения отчетного документа.

2.3. Сведения о функциональных ограничениях

«Технологическая платформа» может быть использована только в среде ОС «Astra Linux SE» версии 1.5 и 1.6, после установки на АРМ DVD-R «Технологической платформы» с дистрибутивом.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Описание алгоритма

Функционирование «Технологической платформы» осуществляется по следующему алгоритму:

- запуск программы и ожидание входных параметров;
- получение параметров из командной строки;
- получение параметров из внешней программы;
- формирование отчета в соответствии с параметрами;
- завершение работы программы.

Алгоритм функционирования «Технологической платформы» в виде блок-схемы изображен на рисунке (Рис. 1).

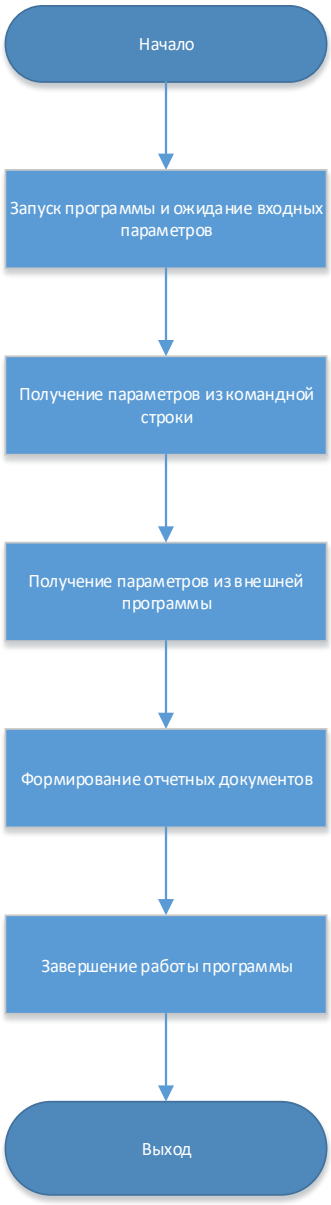

Рис. 1

3.2. Используемые методы

«Технологическая платформа» формирует отчеты на основе шаблона, который представляет собой xml-файл и имеет расширение «.rptdesign». Корневым элементом шаблона является элемент report:

```
<?xml version="1.0" encoding="UTF-8"?>
<report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
    ...
</report>
```

Отчет формируется на основе элементов отчета, которые описаны в шаблоне. Элементы отчета могут быть визуальными (те, что в итоге отображаются в отчете) и невизуальными (имеют вспомогательную роль, например, источники данных). На текущий момент доступны следующие визуальные элементы:

- Label – «Текстовая метка»;
- Text – «Текст»;
- Dynamic Text – «Динамический текст»;
- Data – «Текстовые данные»;
- Grid – «Статическая таблица»;
- List – «Список»;
- Table – «Динамическая таблица».

Все эти элементы располагаются в дочернем по отношению к отчету элементе `<body>`.

«Технологическая платформа» может использовать различные источники данных для формирования отчетов. Все источники данных описываются в элементе отчета <data-sources> с помощью дочерних элементов <oda-data-source>. В настоящем документе приводится описание следующих источников данных: базы данных, файлов в формате CSV. Тип источника данных определяется значением атрибута extensionID. Каждый источник данных должен иметь уникальное значение атрибута name, которое используется для связи с соответствующим источником данных.

3.2.1. Источник данных на основе файла формата CSV

Тип данного источника определяется значением org.eclipse.datatools.connectivity.oda.flatfile атрибута extensionID. Настройки источника данных описываются с помощью дочерних элементов с указанием наименования свойства с помощью атрибута name. Перечень основных свойств представлен в табилце 1.

Таблица 1 – Перечень основных свойств

Свойство	**Тип/Формат/Допустимые значения**	**Описание**
property [@name="HOME"]	Строка	Путь до файла формата CSV в локальной файловой системе

Продолжение таблицы 1

Свойство	Тип/Формат/Допустимые значения	Описание
property [@name="URI"]	Строка	Ссылка на ресурс в сети с файлом формата CSV
property [@name="INCLCOLUMNNAME"]	Булево значение, может принимать значение yes или no	Значение yes указывает на то, что файл содержит в первой строке заголовки колонок (столбцов), при значении no считается, что файл не содержит заголовков столбцов
property [@name="INCLTYPELINE"]	Формат соответствует формату INCLCOLUMNNAME	Определяет, содержит ли вторая строка описание типов столбцов
property [@name="DELIMTYPE"]	Строка, набор допустимых значений: COMMA – запятая «,»; SEMICOLON – точка с запятой «;»; PIPE – вертикальная линия «|»; TAB – символ табуляции	Символ-разделитель значений столбцов. По умолчанию используется значение «COMMA»
property [@name="CHARSET"]	Строка	Кодировка содержимого CSV-файла
property [@name="TRAILNULLCOLS"]	Формат соответствует формату INCLCOLUMNNAME	Значение yes указывает, что для отсутствующих оконечных столбцов в строке должно подставляться пустое значение. При значении no такого происходить не будет. И если в какой-либо строке окажется столбцов меньше, то будет сгенерирована ошибка. Значение по умолчанию: no

Пример описания источника данных:

```
    <report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
        ...
        <data-sources>
            <oda-data-source
extensionID="org.eclipse.datatools.connectivity.oda.flatfile"
name="CSV" id="4">
                <property name="HOME">Данные/</property>
                <property name="DELIMTYPE">SEMICOLON</property>
                <property name="CHARSET">UTF-8</property>
                <property name="INCLCOLUMNNAME">YES</property>
                <property name="INCLTYPELINE">NO</property>
                <property name="TRAILNULLCOLS">NO</property>
            </oda-data-source>
        </data-sources>
        ...
    </report>
```

3.2.2. Источник данных на основе данных из БД

Тип данного источника определяется значением org.eclipse.birt.report.data.oda.jdbc атрибута extensionID. Настройки источника данных описываются с помощью дочерних элементов с указанием наименования свойства с помощью атрибута name. Перечень основных свойств представлен в таблице 2.

Таблица 2 – Перечень основных свойств

Свойство	**Тип/Формат/Допустимые значения**	**Описание**
property[@name="odaDriverClass"]	Строка	Наименование класса JDBC-драйвера для доступа к БД
property[@name="odaURL"]	Строка	Строка соединения с БД. Формат строки зависит от используемого JDBC-драйвера
property[@name="odaUser"]	Строка	Имя пользователя, под учетной записью которого будет осуществляться доступ к БД
encrypted-property[@name="odaPassword"]	Кодированная строка, формат кодирования определяется значением атрибута @encryptionID	Пароль пользователя для доступа к БД. Является закодированным значением

Пример описания источника данных:

```
    <report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
        ...
        <data-sources>
            ...
            <oda-data-source
extensionID="org.eclipse.birt.report.data.oda.jdbc" name="Data Source">
                <property
name="odaDriverClass">org.postgresql.Driver</property>
                <property
name="odaURL">jdbc:postgresql:postgres</property>
            <property name="odaUser">postgres</property>
                <encrypted-property name="odaPassword"
encryptionID="base64">cG9zdGdyZXM=</encrypted-property>
            </oda-data-source>
            ...
        </data-sources>
        ...
    </report>
```

3.3. Программный интерфейс приложений, функционирующих в среде «Технологической платформы»

Программный интерфейс в среде «Технологической платформы» осуществляется виртуальной машиной Java (Java Virtual Machine, далее – JVM), которая является ключевым компонентом платформы Java и описывается в соответствующей спецификации The Java Virtual Machine Specification. JVM доступны для многих аппаратных и программных платформ, поэтому Java может рассматриваться и как связующее программное обеспечение, и как самостоятельная платформа. Таким образом, использование байт-кода позволяет обеспечить независимость Java-программ от целевой операционной системы и аппаратной платформы.

JVM является частью Java Runtime Environment (JRE), среды выполнения для Java-программ, которая также содержит в себе библиотеку Java-классов – Java API. Это стандартный набор модулей, классов, интерфейсов, доступных для любой Java-программы. Библиотека содержит основные алгоритмы и структуры данных, обеспечивающие:

- работу с коллекциями объектов;
- работу с файлами;
- ввод-вывод данных на терминал;
- конвертацию данных между типами;
- математические операции;
- работу с сетью;
- обработку исключений и ошибок в программе;
- поддержку многопоточности.

Программы, написанные на Java и предназначенные для исполнения в JVM, должны быть скомпилированы в стандартизированном переносимом двоичном формате – байт-код, который представляется в виде class-файлов. Программа может состоять из множества классов, размещенных в различных файлах. Для облегчения размещения больших программ, множество class-файлов могут быть упакованы вместе в jar-файлы (Java Archive).

Байт-код Java исполняется внутри экземпляра JVM. За процесс считывания и анализа байт-кода и построение на его основе объектов (экземпляров класса java.lang.Class) отвечает подсистема Class Loader Subsystem, которая состоит из следующих загрузчиков классов (class loader):

- **Bootstrap class loader**: Создается при запуске экземпляра JVM. Он загружает Java API. В отличие от других загрузчиков классов он реализован на нативном коде;
- **Extension class loader**: Загружает классы для расширения API, за исключением базовых классов из Java API. Он также загружает различные функции расширения безопасности;
- **System class loader**: Загружает классы приложений, указанные в параметре CLASSPATH;
- **User-defined class loader**: Это загрузчик классов, который пользователь может создать непосредственно в коде.

Процесс загрузки классов разбит на три фазы:

1) **загрузка (Loading)**: данные из class-файла загружаются в память экземпляра JVM;
2) **связывание (Linking)** – реализуются следующие действия:

– проверка (Verifying): проверяется корректность class-файла и соответствие его спецификации The Java Language Specification,

– подготовка (Preparing): выделяется память экземпляра JVM для структур данных, которые необходимы классу, и определяются поля, методы, интерфейсы, указанные для класса,

– сопоставление (Resolving): опциональное действие, при котором символические ссылки на класс заменяются прямыми;

3) **инициализация (Initializing)**: выполняются блоки статической инициализации, инициируются статические поля класса.

Java поддерживает функцию динамической загрузки классов. Загрузка и связывание класса, когда он ссылается на другой класс, осуществляется во время исполнения, а не во время компиляции. При этом Class Loader Subsystem имеет следующие особенности:

– **иерархическая структура**: загрузчики классов организованы в иерархию с отношениями родитель-потомок. Bootstrap Class Loader является родителем для всех остальных загрузчиков классов;

– **режим делегирования**: на основе иерархической структуры загрузка делегируется между загрузчиками классов. Если загрузка класса была ранее произведена выше по иерархии загрузчиков, то используется уже загруженный класс, иначе выполняется загрузка текущим загрузчиком классов;

– **ограничение видимости**: загрузчик классов может найти класс в родительском загрузчике классов, однако родительский загрузчик классов не может найти класс в дочернем загрузчике классов;

– **выгрузка не допускается**: загрузчик классов может загружать класс, но не может его выгрузить. Вместо выгрузки классов можно удалить текущий загрузчик классов и создать новый загрузчик классов.

Каждый загрузчик классов имеет свое пространство имен, в котором хранятся загруженные классы. Когда загрузчик классов загружает класс, он ищет класс на основе FQCN (Fully Qualified Class Name), хранящегося в пространстве имен, чтобы проверить, был ли класс уже загружен. Даже если класс имеет идентичный FQCN, но другое пространство имен, он рассматривается как другой класс. Другое пространство имен означает, что класс был загружен другим загрузчиком классов.

Загруженная Java-программа размещается в памяти экземпляра JVM. Память раздела на шесть областей (Runtime Data Areas):

– **PC Registers**: хранит адреса текущих инструкций потока. У каждого потока своя область PC Registers;

– **JVM Stack**: хранит стек вызовов методов. Для каждого потока создается своя область JVM Stack. Каждый блок этого стека называется Stack Frame. В этом блоке хранятся контекст вызова метода. После того как поток завершает свою работу, выделенный стек уничтожается;

– **Native method stacks**: эта область используется при вызове нативных методов;

– **Method Area**: в данной области находится вся информация о классах: имена классов, имена родительских классов, методы и поля, а так же прочая информация, включая статические переменные. Существует всего одна область Method Area в экземпляре JVM, и эта область является общим ресурсом для всех потоков, работающих в этом экземпляре JVM;

– **Runtime constant pool**: хранит все ссылки на методы и поля классов, а также значения констант. Эта область включена в область Method Area, однако, она играет важную роль в работе JVM. Поэтому спецификация JVM отдельно описывает ее. Когда осуществляется запрос к константе или обращение к методу или полю класса, экземпляр JVM использует Runtime constant pool и ищет фактический адрес метода или поля;

– **Heap**: хранит объекты (экземпляры классов). Для одного экземпляра JVM существует всего одна область Heap. Она используется всеми потоками, которые работают в этом экземпляре JVM.

Исполнение байт-кода, который был загружен посредством Class Loader Subsystem, осуществляется в механизме Execution engine. Он считывает байт-код последовательно, строчка за строчкой, использует информацию, которая находится в разных областях памяти экземпляра JVM, и выполняет инструкции кода. При этом выполнение может осуществляться следующими путями:

– **в режиме интерпретатора (Interpreter)**: интерпретирует байт-код в команды и выполняет их;

– **в режиме компилятора (Just-In-Time Compiler, JIT)**: компилирует байт-код и преобразует его в нативный код.

Компилятору JIT требуется больше времени для компиляции всего байт-кода, чем для интерпретатора при последовательном исполнении инструкций кода. Поэтому, если байт-код должен быть выполнен только один раз, лучше его интерпретировать вместо компиляции. JVM проверяет, как часто выполняется конкретная часть байт-кода, и компилируют только тогда, когда частота выше определенного уровня.

Во время исполнения Java-программ происходит создание объектов. Необходимость объектов определяется логикой программы. В дальнейшем процессе исполнения программы необходимость в некоторых объектах отпадает, и требуется освободить занятую ими память. Для этого в JVM имеется специальный механизм – сборщик мусора (Garbage Collector). Он работает в отдельном параллельном потоке и маркирует неиспользуемые объекты с целью освобождения и повторного использования занятых ими областей памяти.

3.4. Структура программы

«Технологическая платформа» реализована как самостоятельная программа и не делится на составные части.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Для функционирования «Технологической платформы» необходим персональный компьютер со следующей конфигурацией:

1) двухъядерный (2 core) процессор с архитектурой AMD64, рабочая частота (не в turbo-режиме) ядра не менее 2 ГГц;

2) оперативная память объемом не менее 4 Гбайт;

3) НЖМД с объемом памяти не менее 40 Гбайт.

5. ВЫЗОВ И ЗАГРУЗКА

5.1. Запуск программы

Запуск «Технологической платформы» для формирования отчетных документов производится путем выполнения следующей команды в директории ./usr/bin/:

```
/usr/bin/report-generator
```

В случае успешного выполнения команды на экране отобразится сообщение следующего вида:

```
Help for ReportRunner
--mode/-m [run|render|runrender] [options] [rptdesign|rptdocument]
        The default mode is runrender.
To see options for run mode, use:
        --help/-h run
To see options for render mode, use:
        --help/-h render
To see options for runrender mode, use:
        --help/-h runrender
Print current message, use --help/-h
```

Примечание. Перечень параметров «Технологической платформы», которые могут быть использованы при запуске, приведены в приложении к документу «Технологическая платформа. Руководство системного программиста» НПБК.20765-01 32 01.

5.2. Необходимые пакеты для установки «Технологической платформы»

Перед установкой «Технологической платформы» необходимо установить пакеты *libnss3* из состава ОС «Astra Linux SE» версии 1.5 или 1.6.

5.3. Установка «Технологической платформы»

Для установки «Технологической платформы» для Astra Linux версии 1.5 необходимо в консоли с правами суперпользователя выполнить команду:

```
dpkg -i openjdk-8-jre_8u141-b15-3_amd64.deb
dpkg -i openjdk-8-jre-headless_8u141-b15-3_amd64.deb
dpkg -i report-generator_4.6.0_amd64.deb
```

Для установки «Технологической платформы» для Astra Linux версии 1.6 необходимо в консоли с правами суперпользователя выполнить команды:

```
dpkg -i openjdk-11-jre-headless_11.0.8+10-1_amd64.deb
dpkg -i openjdk-11-jre_11.0.8+10-1_amd64.deb
dpkg -i report-generator_4.9.0_amd64.deb
```

5.4. Проверка установки

Для того чтобы выполнить проверку установки «Технологической платформы» необходимо в консоли выполнить команду:

```
java –version
```

При успешной установке «Технологической платформы», на экране должно отобразиться сообщение следующего вида:

```
openjdk version "11.0.4" 2019-07-16
OpenJDK Runtime Environment (build 11.0.4+11-post-AstraLinuxSE-1bpo91)
OpenJDK 64-Bit Server VM (build 11.0.4+11-post-AstraLinuxSE-1bpo91, mixed mode, sharing)
```

5.5. Контрольные суммы текстов программы на исходном языке

Основной характристикой файлов, входящих в «Технологическую платформу» является контрольная сумма информации, подсчитанная с помощью программы «md5sum» из состава ОС «Astra Linux SE» версии 1.5 или 1.6. Контрольная сумма текста программы на исходном языке приведена пофайлово в таблице (Таблица 3).

Таблица 3 – Контрольные суммы текстов программы на исходном языке

Наименование файла	**Контрольная сумма**
\AL_1_5\report-generator-build.tgz	75fdddad4437f443f8ad92e3c756495e
\AL_1_6\tpo-sources.tar	99f267aaf3847bb020b49bf6311bef18

6. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

6.1. Входные данные

Входные данные для «Технологиечской платформы» определяются наборами данных. Каждый набор данных определяет, какие данные будут доступны для отчета из источников данных. В соответствии с типами источников данных будет рассмотрено два типа наборов данных: набор данных из файла формата CSV и набор данных из базы данных. Все наборы данных описываются в элементе отчета <data-sets> с помощью дочерних элементов <oda-data-set>. Тип набора данных определяется значением атрибута extensionID. Каждый набор данных должен иметь уникальное значение атрибута name, которое используется для связи с соответствующим набором данных. Для связи набора данных с источником данных используется свойство (дочерний элемент <property>) со значением атрибута name="dataSource".

У набора данных должна быть описана дочерняя структура, которая представляет из себя дочерний элемент <list-property name="resultSet">. У данной дочерней структуры должны быть описаны колонки по количеству, которое соответствует количеству полей, доступных из запроса. Каждое поле описывается элементом <structure> и содержит перечень основных свойств, представленный в таблице 4.

Таблица 4 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="position"]	Целое число, 1..N	Позиция колонки в наборе данных
property[@name="name"]	Строка	Системное имя колонки в наборе данных
property[@name="dataType"]	BindingDataType	Тип значения
property[@name="nativeName"]	Строка	Наименование поля в источнике данных
property[@name="nativeDataType"]	Целое число. Допустимые значения: 1) 4 – integer; 2) 12 – varchar и пр.	Тип поля в источнике данных

У набора данных может быть описана дочерняя структура <list-property name="columnHints">, которая дополняет описание выбранных колонок. У данной дочерней структуры должны быть описаны колонки по количеству, которое соответствует количеству полей, доступных из запроса. Каждое поле описывается элементом <structure> и содержит перечень основных свойств, представленный в таблице 5.

Таблица 5 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
<property name="columnName">	Строка	Системное имя колонки в наборе данных

Продолжение таблицы 5

Свойство	Тип/Формат/Допустимые значения	Описание
<property name="analysis">	Строка, может принимать только следующие значения: 1) measure – для оценочно/характеристических колонок; 2) dimension – для описательных колонок	Маркер типа значения
<text-property name="heading">	Строка	Наименование колонки для отображения

У набора данных может быть описана дочерняя структура <list-property name="computedColumns">, которая описывает вычисляемые колонки. Каждое поле описывается элементом <structure> и содержит перечень основных свойств, преставленный в таблице 6.

Таблица 6 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="name"]	Строка	Системное имя колонки
property[@name="dataType"]	BindingDataType	Тип значения
expression[@name="expression"]	JSExpression	Функция для вычисления значения по другим поля строки таблицы

6.1.1. Набор данных на основе файла формата CSV

Тип данного набора данных определяется значением org.eclipse.datatools.connectivity.oda.flatfile.dataSet атрибута extensionID. Настройки набора данных описываются с помощью дочерних элементов с указанием наименования свойства с помощью атрибута name. Перечень основных свойств представлен в таблице 7.

Таблица 7 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="dataSource"]	Строка	Наименование источника данных
property[@name="rowFetchLimit"]	Целое число	Максимальное количество строк источника данных, которые будут доступны с помощью данного набора данных. По умолчанию ограничения нет
xml-property[@name="queryText"]	Строка (используется элемент CDATA)	Запрос на получение данных

Пример описания набора данных:

```
    <report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
        ...
        <data-sets>
            <oda-data-set
extensionID="org.eclipse.datatools.connectivity.oda.flatfile.dataSet"
name="Продукты" id="109">
```

```
                <list-property name="computedColumns">
                    <structure>
                        <property name="name">Сумма за пункт</property>
                        <expression
name="expression">row["Количество"]*row["Цена"]</expression>
                        <property name="dataType">decimal</property>
                    </structure>
                </list-property>
                <list-property name="columnHints">
                    <structure>
                    <property name="columnName">№ п/п</property>
                        <property name="analysis">dimension</property>
                        <text-property name="heading">№ п/п</text-
property>
                    </structure>
                    <structure>
                        <property
name="columnName">Наименование</property>
                        <property name="analysis">dimension</property>
                        <text-property
name="heading">Наименование</text-property>
                    </structure>
                    <structure>
                        <property
name="columnName">Количество</property>
                        <property name="analysis">measure</property>
                        <text-property name="heading">Количество</text-
property>
                    </structure>
                    <structure>
                        <property name="columnName">Цена</property>
                        <property name="analysis">measure</property>
                        <text-property name="heading">Цена</text-
property>
                    </structure>
                </list-property>
                <property name="dataSource">CSV</property>
                <list-property name="resultSet">
                    <structure>
                        <property name="position">1</property>
                        <property name="name">№ п/п</property>
                        <property name="nativeName">№ п/п</property>
                        <property name="dataType">string</property>
                        <property name="nativeDataType">12</property>
                    </structure>
```

```
                    <structure>
                        <property name="position">2</property>
                        <property name="name">Наименование</property>
                        <property
name="nativeName">Наименование</property>
                        <property name="dataType">string</property>
                        <property name="nativeDataType">12</property>
                    </structure>
                    <structure>
                        <property name="position">3</property>
                        <property name="name">Количество</property>
                        <property
name="nativeName">Количество</property>
                        <property name="dataType">decimal</property>
                        <property name="nativeDataType">2</property>
                    </structure>
                    <structure>
                        <property name="position">4</property>
                        <property name="name">Цена</property>
                        <property name="nativeName">Цена</property>
                        <property name="dataType">decimal</property>
                        <property name="nativeDataType">2</property>
                    </structure>
                </list-property>
                <xml-property name="queryText"><![CDATA[select "№ п/п",
"Наименование", "Количество", "Цена" from "Продукты.csv" : {"№ п\/п","№
п\/п",STRING;"Наименование","Наименование",STRING;"Количество","Количес
тво",BIGDECIMAL;"Цена","Цена",BIGDECIMAL}]]></xml-property>
            </oda-data-set>
        </data-sets>
            ...
    </report>
```

6.1.2. Набор данных на основе данных из БД

Тип данного набора данных определяется значением org.eclipse.birt.report.data.oda.jdbc.JdbcSelectDataSet атрибута extensionID. Настройки набора данных описываются с помощью дочерних элементов <property> с указанием наименования свойства с помощью атрибута name. Перечень основных свойств представлен в таблице 8:

Таблица 8 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="dataSource"]	Строка	Наименование источника данных
xml-property[@name="queryText"]	Строка (используется элемент CDATA)	Запрос на получение данных

Пример описания набора данных:

```
    <report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
        ...
        <data-sets>
            <oda-data-set
extensionID="org.eclipse.birt.report.data.oda.jdbc.JdbcSelectDataSet"
name="Data Set">
                <property name="dataSource">Data Source</property>
                <list-property name="resultSet">
                    <structure>
                    <property name="position">1</property>
                        <property name="name">id</property>
                        <property name="nativeName">id</property>
                        <property name="dataType">integer</property>
                        <property name="nativeDataType">4</property>
                    </structure>
                    <structure>
                        <property name="position">2</property>
                        <property name="name">firstname</property>
                        <property
name="nativeName">firstname</property>
                        <property name="dataType">string</property>
                        <property name="nativeDataType">12</property>
                    </structure>
                    <structure>
                        <property name="position">3</property>
                        <property name="name">middlename</property>
                        <property
name="nativeName">middlename</property>
                        <property name="dataType">string</property>
                        <property name="nativeDataType">12</property>
                    </structure>
                    <structure>
                        <property name="position">4</property>
                        <property name="name">lastname</property>
                        <property name="nativeName">lastname</property>
                        <property name="dataType">string</property>
                        <property name="nativeDataType">12</property>
                    </structure>
                </list-property>
                <xml-property name="queryText"><![CDATA[select * from
public.people]]></xml-property>
            </oda-data-set>
            ...
    </report>
```

6.2. Выходные данные

Выходными данными «Технологической платформы» являются отчеты, сформированные в соответствии с указанными параметрами. Параметры отчета используются при формировании условий выборки и настройки отображения данных. Все параметры отчета описываются в элементе отчета <parameters> с помощью дочерних элементов <scalar-parameter>. Другие настройки параметров отчета представлены в таблице 9.

Таблица 9 – Настройки парамтеров отчета

Свойство	Тип/Формат/Допустимые значения	Описание
@name	Строка	Системное имя параметра
text-property[@name="promptText"]	Строка	Имя параметра для отображения
property[@name="valueType"]	Строка, может принимать только следующие значения: 1) static – значение вводится пользователем или выбирается из фиксированного списка; 2) dynamic – значение вводится пользователем или выбирается из динамики формируемого списка	Способ получения/формирования значения
property[@name="dataType"]	BindingDataType	Тип значения
property[@name="isRequired"]	Строка (формат xs:boolean)	Маркер обязательности
property[@name="hidden"]	Строка (формат xs:boolean)	Маркер видимости при формировании условий запроса отчета
property[@name="paramType"]	Строка со значением «simple»	Тип параметра
property[@name="controlType"]	Строка со значением «text-box»	Визуальный элемент управления для формирования значения
structure[@name="format"]	FormatStructure	Формат значения
simple-property-list[@name="defaultValue"]/ value[@type="constant"]	Строка	Значение по умолчанию

Пример описания параметра отчета:

```
    <report xmlns="http://www.eclipse.org/birt/2005/design"
version="3.2.23">
        ...
        <parameters>
            <scalar-parameter name="pYear" id="68">
                <text-property name="promptText">Год</text-property>
                <property name="valueType">static</property>
                <property name="isRequired">true</property>
                <property name="dataType">date</property>
                <property name="distinct">true</property>
                <list-property name="selectionList"/>
                <property name="paramType">simple</property>
                <property name="controlType">text-box</property>
```

```
            <structure name="format">
                <property name="category">Custom</property>
                <property name="pattern">yyyy</property>
            </structure>
        </scalar-parameter>
    </parameters>
    ...
</report>
```

Помимо самих элементов в отчете описываются макеты страниц, на которых отображаются элементы. Для этого в отчете добавляется элемент <page-setup>, дочерние элементы которого <simple-master-page> и описывают макеты. Каждый макет страницы должен иметь атрибут name, значение которого используется для ссылки на данный макет. Значение данного атрибута у всех макетов страниц должно быть уникальным.

Настройки макета страницы описываются с помощью дочерних элементов с указанием наименования свойства с помощью атрибута name. Перечень основных свойств представлен в таблице 10.

Таблица 10 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="type"]	Строка, может принимать следующие значения: – a3 соответствует формату бумаги A3; – a4 соответствует формату бумаги A4; – a5 соответствует формату бумаги A5; – custom позволяет задать произвольный размер страницы (используются свойства width и height)	Формат страницы
property[@name="width"]	MultiformatSize	Ширина страницы
property[@name="height"]	MultiformatSize	Высота страницы
property[@name="orientation"]	Строка, принимающая одно из следующих значений: 1) landscape – горизонтальная ориентация страницы; 2) portrait – вертикальная ориентация страницы	Ориентация страницы
property[@name="backgroundColor"]	ColorNameOrRGB	Цвет фона страницы
property[@name="headerHeight"]	MultiformatSize	Высота верхнего колонтитула. Значение по умолчанию: 0.5in.
property[@name="footerHeight"]	MultiformatSize	Высота нижнего колонтитула. Значение по умолчанию: 0.5in.

Пример описания макета страницы:

```
<simple-master-page name="Simple MasterPage">
    <property name="type">custom</property>
    <property name="orientation">portrait</property>
```

```
    <property name="backgroundColor">teal</property>
    <property name="height">210mm</property>
    <property name="width">420mm</property>
    <property name="headerHeight">1.2cm</property>
    <property name="footerHeight">1.2cm</property>
</simple-master-page>
```

Для описания верхнего и нижнего колонтитулов используются два элемента:

1) <page-header> – для верхнего колонтитула;
2) <page-footer> – для нижнего колонтитула.

Пример описания приведен ниже:

```
<page-setup>
    <simple-master-page name="Simple MasterPage">
        <page-header>
            ...
        </page-header>
        <page-footer>
            ...
        </page-footer>
    </simple-master-page>
</page-setup>
```

Как у верхнего, так и у нижнего колонтитула возможен только один дочерний элемент в зависимости от того, какую информацию требуется отображать в колонтитулах. В качестве дочернего элемента может выступать либо любой визуальный элемент, либо элемент автотекста.

Элемент автотекста определяется с помощью элемента <auto-text>. Дополнительные настройки элемента автотекста описываются с помощью дочерних элементов с указанием наименования свойства с помощью атрибута name. Перечень основных свойств приведен в таблице 11.

Таблица 11 – Перечень основных свойств

Свойство	Тип/Формат/Допустимые значения	Описание
property[@name="type"]	Строка, одно из значений: 1) page-number – номер страницы; 2) total-page – всего страниц	Тип элемента автотекста

Доступны также и другие свойства для настройки визуального представления автотекста, такие как «Общие настройки», «Настройки отступа для содержимого», «Настройки отступа для границ», «Настройки отображения границ», «Разметки страницы», «Условие отображения».

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

НЖМД – накопитель на жестких магнитных дисках;
ОС – операционная система;
ПИ – программное изделие.

Лист регистрации изменений

Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ докумен-та*	*Входящий № сопрово-дительно-го докум. и дата*	*Подп.*	*Да-та*
Изм.	*изменен-ных*	*заменен-ных*	*новых*	*аннули-рован-ных*					
1	–	Все	–	–	26	НПБК.299-2017			
2	–	Все	–	–	26	НПБК.427-2019			
3	–	Все	–	–	25	НПБК.594-2021			
4	–	Все	–	–	25	НПБК.646-2021			
5	–	Все	–	–	25	НПБК.718-2022			