ARTICLE HISTORY: Received: September 20, 2021 Accepted: November 13, 2021 Published: November 19, 2021

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПОДСИСТЕМ АСИНХРОННОЙ ГЕНЕРАЦИИ ОТЧЕТНОСТИ В ИНФОРМАЦИОННЫХ СИСТЕМАХ.

Плетнев Андрей Владимирович
Независимый исследователь,
Директор ТОО «SimCo Soft»,
Руководитель группы разработки ТОО «OneBill»,
Республика Казахстан, г. Алматы.

DESIGN AND IMPLEMENTATION OF ASYNCHRONOUS REPORT GENERATING SUBSYSTEMS IN INFORMATION SYSTEMS.

Pletnev Andrey Vladimirovich
Independent researcher,
CEO «SimCo Soft» LLP,
Team lead «OneBill» LLP,
Republic of Kazakhstan, Almaty city.

Аннотация. Данная статья предлагает вниманию читателя описание методологии проектирования и реализации подсистем асинхронной генерации отчетов на основе шаблонов для клиент-серверных Информационных систем, построенных с применением web-технологий. В статье даются пошаговые рекомендации: от организации базовой структуры шаблонов отчетов до интеграции готового генератора отчетов в Информационную систему.

Abstract. This article offers the reader a description of the methodology for the design and implementation of asynchronous reporting subsystems based on templates for client-server Information Systems built using web technologies. The article provides step-by-step recommendations: from organizing the basic structure of report templates to integrating a ready-made report generator into the Information System.

Ключевые слова: web-технологии, отчеты, шаблоны, frontend, backend.

Keywords: web-technologies, reports, templates, frontend, backend.

Введение.

Несмотря на то, что в данное время существует некоторое количество готовых решений для генерации отчетности, встраиваемых в Информационные системы, зачастую, по тем или иным причинам, разработчикам программного продукта бывает недостаточно их функциональности для реализации поставленных задач. Кроме этого, такое готовое решение может не работать в технологическом стеке разрабатываемого проекта. В таких ситуациях разработчики сталкиваются с необходимостью реализации собственной подсистемы генерации отчетности для разрабатываемой ими Информационной системы.

В этой статье я хочу поделиться своим опытом проектирования и разработки подсистем асинхронной генерации отчетности в Информационных системах. Материал этой статьи представлен без привязки к какому-то определенному технологическому стеку и будет полезен руководителям отделов и специалистам по разработке программного обеспечения, проектным менеджерам и системным архитекторам. Благодаря данному методологическому руководству, специалисты по разработке программного обеспечения, независимо от используемого ими технологического стека, смогут освоить:

1. Принцип построения и структуру шаблонов отчетов;
2. Принцип работы модуля генерации отчетов: процесс генерации отчета на основе шаблонов;
3. Процесс интеграции модуля генерации отчетов в Информационную систему;
4. Преимущества асинхронного метода взаимодействия Информационной системы с модулем генерации отчетов.

Также, в контексте описываемой темы, я буду давать практические рекомендации разработчикам: на что следует обратить внимание или каким образом должна быть реализована та или иная часть функционала.

Кроме этого, в конце статьи я опишу несколько полезных вспомогательных функций для Вашего будущего модуля генерации отчетов, реализовав которые, Вы существенно улучшите функциональные возможности Вашего продукта и, как следствие, повысите его уникальность и коммерческую ценность.

Структура шаблона отчета.

Процесс генерации отчетности, так или иначе, производится в два этапа – компиляция и экспорт. К подробному описанию этих этапов мы приступим немного позже в разделе «Процесс генерации отчета», а

сейчас, для начала, выясним, что же требуется для выполнения этапа компиляции отчета. Известно, что любой отчет строится по заранее определенным наборам правил. Для компилятора внутри генератора отчетов таким набором правил является шаблон отчета, который должен включать:

1. Определения параметров страницы: размер и ориентация бумаги, отступы и т.д.;
2. Определение состава, типов и описаний для входных параметров отчета;
3. Определения для внутренних переменных отчета;
4. Набор разделов (chapters) отчета. В самом простом случае шаблон отчета будет содержать только один раздел;
5. Определение состава секций данных для раздела;
6. Определение параметров секции данных. Для построения многоуровневых отчетов секции данных могут иметь вложенные секции, образуя иерархическую структуру. Физических ограничений шаблона на глубину вложенности секций быть не должно – компилятор генератора должен просматривать вложенные секции на любую глубину. В самом простом случае построения шаблона отчета никаких вложенных секций не будет – простой одноуровневый отчет;
7. Определение для получаемого набора данных (dataset) секции;
8. Определение состава разнообразных контейнеров полей (fieldset) внутри секции и их атрибутов;
9. Определение состава, типа и атрибутов элементов отображения данных отчета внутри наборов полей секции раздела: текстовые поля и поля данных, диаграммы и графики, графические элементы – их координаты, размеры, параметры шрифта и пр.;

Следуя перечисленным выше требованиям, мы можем построить наглядную схему структуры шаблона отчета, пример которой показан на Рисунке 1.

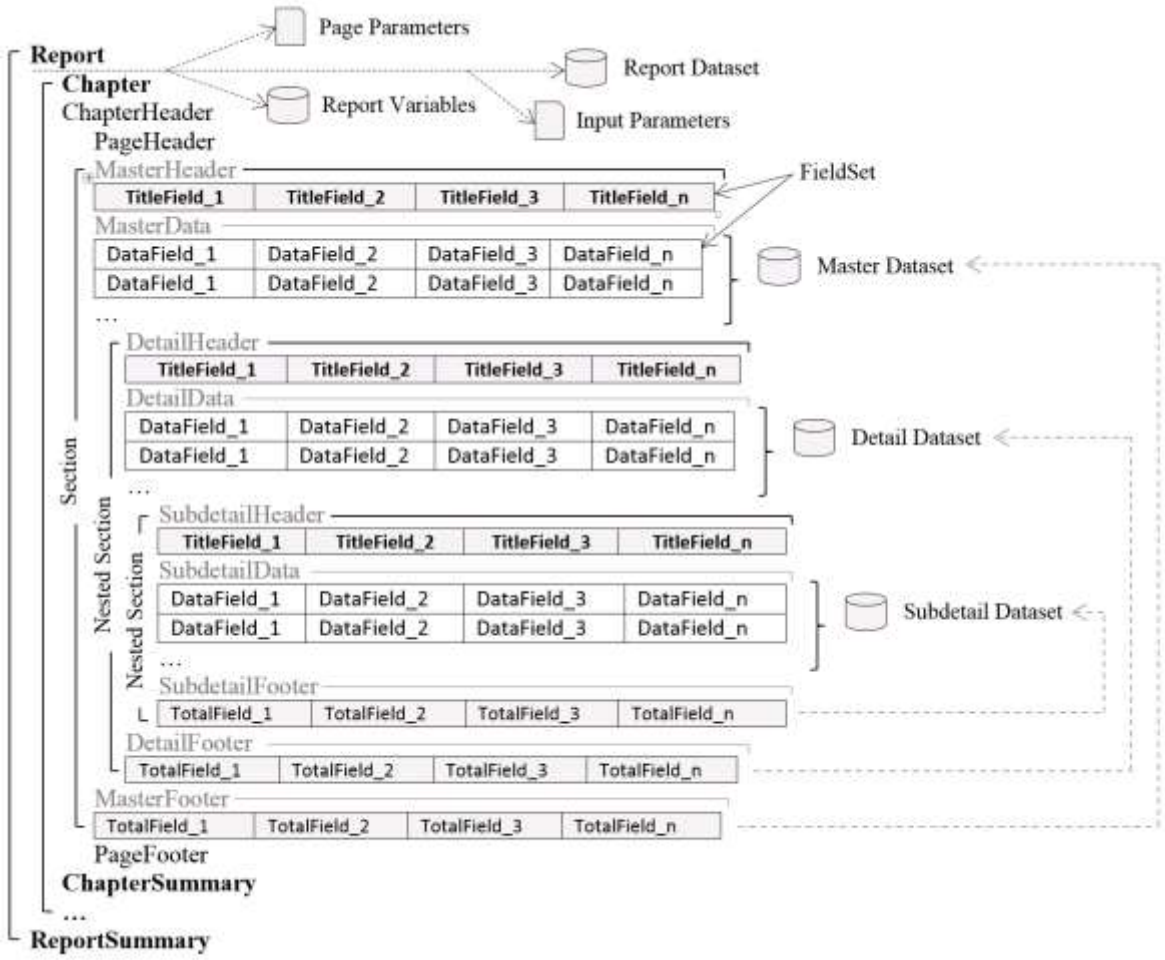

Рисунок 1. «Схема логической структуры шаблона отчета»

Как видно из представленной схемы, шаблон представляет собой иерархическую структуру [3], каждый элемент которой имеет собственный набор атрибутов, описывающих его свойства, и может включать в себя другие вложенные элементы со своими наборами атрибутов, о которых я расскажу подробно далее в этой части

статьи. На первый взгляд, может показаться, что схема, отображенная на Рисунке 1, имеет строго определенную и выстроенную структуру. Отчасти это так – корневой элемент ***Report***, состав его обязательных атрибутов, наличие минимум одного раздела ***Chapter*** и минимум одной секции данных ***Section*** внутри раздела ***Chapter*** обязательны. Однако, на более глубоких уровнях иерархии, строение шаблона отчета становится более слабоструктурированным: одни элементы могут быть необязательными, а состав и разнообразие других может сильно отличаться в разных шаблонах отчетов – все зависит от поставленной задачи для каждого конкретного шаблона отчета. Такая организация структуры дает разработчику шаблонов широчайшие возможности для реализации поставленных задач по построению всего разнообразия отчетов для Информационных систем. Исходя из этого, наиболее подходящим форматом для физического описания комплексной структуры шаблонов отчетов является формат XML [5], который хорошо подходит для описания слабоструктурированных комплексных объектов [3]. Похожим образом выглядит структура web-страниц на языке разметки HTML, когда web-страница содержит множество произвольных элементов, внутри которых в произвольном порядке располагается множество других разнородных элементов. Например, заголовочный текст, абзацы текста, графика, списки, таблицы и т.д. Кроме этого, практически во всех современных языках программирования есть библиотеки для работы с форматом XML.

Давайте же подробнее разберем состав каждого элемента структуры шаблона, показанного на Рисунке 1. Вы можете свободно добавлять атрибуты элементов шаблона, необходимые для решения Ваших задач, а также убирать неиспользуемые и\или бесполезные, на Ваш взгляд, атрибуты для всех описанных ниже элементов:

Report – корневой элемент шаблона отчета. Может включать следующие атрибуты:

Name – имя отчета – строка;

Description – описание шаблона – строка;

CreateDate – дата создания шаблона – строка в формате даты;

Version – версия шаблона – строка;

CVersion – версия компилятора шаблона, требуемая для обработки данного шаблона – строка;

ReleaseDate – дата релиза версии шаблона – строка в формате даты;

Author – имя и информация об авторе – строка.

Помимо основных атрибутов шаблона, корневой элемент ***Report***, согласно представленной схемы, включает в себя другие важные дочерние элементы:

PageParameters – Определения параметров страницы. Значения атрибутов этого элемента будут необходимы генератору отчета на этапе *экспорта* скомпилированного отчета. Элемент имеет следующие атрибуты:

PageWidth – ширина листа бумаги – вещественное;

PageHeight – высота листа бумаги – вещественное;

PageOrientation – ориентация листа – строка: *книжная* или *альбомная.* Разработчик компилятора и конструктора шаблонов для генератора отчета должен позаботиться о согласованности имен хранимых значений;

LeftMargin – ширина левого поля – вещественное;

TopMargin – высота верхнего поля – вещественное;

RightMargin – ширина правого поля – вещественное;

BottomMargin – высота нижнего поля – вещественное.

Endless – признак «бесконечного листа» - логическое. Может быть полезно для систем, где используются принтеры с рулонной подачей бумаги и\или когда разбивка отчета на страницы не нужна.

Стоит отметить, что важно использовать одни и те же единицы измерения, применяемые в шаблоне отчета для определения размеров и координат элементов на листе, размеров шрифтов, отступов, толщины линий и т.д. Если вы приняли как внутренний стандарт использование какой-то определенной единицы измерения (миллиметры, дюймы, пункты), то согласуйте это с разработчиками конструктора шаблонов и модуля *экспорта* в генераторе отчетов. Разработчик модуля *экспорта* для генератора отчетов должен позаботиться о корректном пересчете единиц измерения при выполнении процедуры *экспорта* отчета в разные форматы.

ReportVariables – список пользовательских переменных, которые могут быть необходимы разработчику шаблона для построения отчета. Значения переменных, определенных в данном списке, будут доступны глобально во всем шаблоне, не зависимо от того, на каком уровне иерархии располагается элемент, использующийся для отображения значения данной переменной. Стоит отметить, что значения, определенные в этом списке, также могут быть использованы разработчиком шаблона как константы, что в ряде случаев может быть очень полезно. Элемент ***ReportVariables*** в шаблоне оформляется в виде массива дочерних элементов, каждый из которых является определением отдельной переменной\константы и содержит следующие атрибуты:

Name – имя переменной – строка. Разработчик конструктора шаблонов отчетов должен обеспечить проверку на уникальность имен переменных в рамках списка переменных отчета;

Type – тип переменной – строка (список возможных значений: *int*, *double*, *string*, *date*, *time*, *datetime*). Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений;

DefaultValue – Значение переменной по умолчанию\значение константы – строка.

Использование ***ReportVariables*** в отчете не является обязательным условием и предназначено исключительно для решения особых нужд разработчиков шаблонов отчетов. Таким образом, в некоторых шаблонах отчетов этот элемент может быть пустым массивом – разработчику компилятора шаблона для генератора отчета следует учесть этот факт.

InputParameters – список входных параметров, необходимых для построения отчета. Значения входных параметров задает пользователь Информационной системы на форме создания нового отчета (см. ниже раздел «***Интеграция генератора отчетов в Информационную систему***»). Элемент ***InputParameters*** в шаблоне оформляется в виде массива дочерних элементов, каждый из которых является определением для отдельного входного параметра и содержит следующие атрибуты:

Name – имя параметра – строка. Разработчик конструктора шаблонов отчетов должен обеспечить контроль уникальности имен входных параметров в рамках списка входных параметров отчета;

DisplayLabel – отображаемое название параметра – строка. Текст, который будет отображен над полем ввода значения параметра на форме создания нового отчета в Информационной системе;

DisplaySortOrder – порядок отображения поля ввода данных для этого параметра на форме создания нового отчета в Информационной системе – целое. Разработчик frontend Информационной системы должен обеспечить сортировку полей ввода данных на форме создания нового отчета по возрастанию (располагать на форме – сверху вниз);

Type – тип входного параметра – строка (список возможных значений: *int, bool, double, string, date, time, datetime, date interval, time interval, datetime interval, list*). Определяет тип данных для входного параметра. Разработчик компилятора и конструктора шаблона для генератора отчета, а также разработчик frontend Информационной системы должны позаботиться о согласованности имен возможных значений. Разработчик frontend Информационной системы должен обеспечить отображение поля ввода данных, соответствующего указанному для этого входного параметра типу данных, а также передачу на сервер вводимого пользователем значения;

DefaultValue – значение параметра по умолчанию – строка. Разработчик frontend Информационной системы должен обеспечить отображение этого значения в соответствующем этому входному параметру поле ввода. Для полей типа *list* в качестве значения по умолчанию передается идентификатор элемента в списке, который должен быть выбран по умолчанию;

Dataset – текст запроса к базе данных для получения списка значений для поля ввода данных типа *list* – строка. Результат выполнения запроса к базе данных должен возвращать минимум 2 поля: идентификатор элемента списка и текстовое значение элемента списка: *id* и *name* (если имена соответствующих полей в базе данных отличаются от *id* и *name*, то разработчик шаблона отчета должен применять псевдонимы в запросе). Если при выполнении запроса к базе данных необходимо получить дополнительные поля данных, то имена этих полей уже могут быть произвольными. Разработчик backend Информационной системы, при обработке запроса от frontend на получение списка параметров отчета для формы создания нового отчета, должен обеспечить выполнение запроса к базе данных, когда *Type = list* и выполнить возврат объект списка, полученного в результате выполнения этого запроса. При этом, разработчик frontend Информационной системы должен обеспечить отображение поля ввода типа SELECT и загрузить в него данные из полученного списка;

PassValue – имя поля, значение которого необходимо передать из поля ввода типа *list* с формы создания нового отчета в Информационной системе – строка. Разработчик frontend Информационной системы должен обеспечить передачу значения указанного поля для выбранного элемента списка данных, полученных для поля типа *list*. Разработчик конструктора шаблонов должен обеспечивать установку значения “id” для этого поля по умолчанию, когда разработчик шаблона отчетов в конструкторе шаблонов для поля выбирает тип *list*.

ReportDataset – текст запроса к базе данных – строка. Результат выполнения этого запроса в виде набора значений полей будет доступен глобально во всем отчете. Как правило, такой запрос должен возвращать одну запись либо один комплексный объект. Допускается не использовать ***ReportDataset*** – разработчикам конструктора и компилятора шаблона для генератора отчетов следует учесть этот факт. Способ передачи параметров в запрос к базе данных будет описан ниже в разделе «***Компиляция шаблона***».

На Рисунке 2 представлен пример оформления описанных выше элементов шаблона и их атрибутов в формате XML.

```xml
<?xml version="1.0" encoding="utf-8"?>
<Report Name="Quarterly Sales Report" Description="" CreateDate="01-01-2021" Version="1.2.3"
CVersion="1.0.0" ReleaseDate="22-03-2021" Author="Andrey Pletnev">
  <PageParams PageWidth="210.0" PageHeight="297.0" PageOrientation="vertical" LeftMargin="20.0"
  TopMargin="20.0" RightMargin="20.0" BottomMargin="20.0" Endless="false" />
  <ReportVariables>
    <Variable Name="UserVar1" Type="int" DefaultValue="0" />
    <Variable Name="UserVar2" Type="bool" DefaultValue="false" />
    <Variable Name="UserVar3" Type="string" DefaultValue="" />
  </ReportVariables>
  <InputParams>
    <Param Name="dt" DisplayLabel="Select date" DisplaySortOrder="0" Type="date"
    DefaultValue="=now()" DataSet="" PassValue="" />
    <Param Name="dep" DisplayLabel="Select department" DisplaySortOrder="1" Type="list"
    DefaultValue="1" DataSet="SELECT id, name FROM departments ORDER BY id" PassValue="id" />
  </InputParams>
  <ReportDataset>
    <![CDATA[
      SELECT id, name, date, params FROM user_report_params
      WHERE user_id = @uid
    ]]>
  </ReportDataset>
  <Chapters>
    ...
  </Chapters>
</Report>
```

Рисунок 2. «Представление элементов шаблона отчета в формате XML»

Теперь, когда мы изучили основные элементы шаблона отчета и их свойства, давайте перейдем к описанию внутренней структуры шаблона отчета, а именно той его части, которая отвечает за описание структуры *разделов* (***Chapters***) отчета.

Считаю необходимым немного нарушить последовательность описания иерархии элементов шаблона отчета и, забегая вперед, сначала рассказать о таком важном типе элементов отчета как ***FieldSet***. Так как этот тип является общеупотребимой структурной единицей для многих элементов шаблона отчета, то ссылаться на уже описанный тут тип ***FieldSet*** будет проще для дальнейшего понимания материала.

Итак, ***FieldSet*** не является конкретным элементом структуры шаблона отчета, но представляет собой определение типа элементов шаблона отчета, которые могут содержать в себе множество разнородных *элементов отображения данных*. Это та слабоструктурированная часть шаблона отчета, о которой я упоминал выше. На Рисунке 1 такие элементы *отображения данных* показаны как элементы с надписями ***TitleField***, ***DataField*** и ***TotalField*** – в данном примере это элементы отображения текста. Говоря о разнородных *элементах отображения данных*, я имею в виду следующие возможные типы элементов:

TextView – элемент отображения текста (текст, числа, даты);

ImageView – элемент отображения изображений (иконки, значки, логотипы, фотографии);

BarcodeView – элемент для отображения 1D и 2D штрих-кодов;

ChartView – элемент отображения графиков и диаграмм;

LineView – элемент отображения линии;

ShapeView – элемент отображения геометрической фигуры.

Если, с точки зрения академического определения типов структур данных, структура шаблона отчета является «деревом», то *элементы отображения данных* являются «листьями» этого «дерева» [3]. Это означает, что *элементы отображения данных* не могут\не должны иметь дочерних элементов. Кроме этого, ***FieldSet*** также может быть и пустым, т.е. не содержать в себе никаких элементов – тут все зависит от задач, решаемых разработчиком шаблона отчета. Разработчикам компилятора и конструктора шаблона для генератора отчета следует учесть данные обстоятельства.

Как и все описанные выше элементы шаблона отчета, *элементы отображения данных* имеют свой набор атрибутов. Нижеперечисленный набор атрибутов есть у всех типов *элементов отображения данных*:

Name – уникальное имя элемента – строка. Разработчик конструктора шаблонов отчетов должен обеспечить контроль уникальности имен элементов в рамках всего шаблона отчета;

Left – положение элемента относительно левого края контейнера, в котором расположен данный элемент – вещественное;

Top – положение элемента относительно верхнего края контейнера, в котором расположен данный элемент – вещественное;

Height – высота элемента – вещественное;

Width – ширина элемента – вещественное;

Color – цвет заливки элемента – строка. HEX-представление цвета RGB. (*none* – отсутствие заливки, прозрачный фон);

FrameColor – цвет линий – строка. HEX-представление цвета RGB. Для элементов типа *TextView*, *ImageView*, *BarcodeView* и *ChartView* определяет цвет линий рамки вокруг элемента, определенных в свойстве *FrameEdges*. Для элементов типа *LineView* и *ShapeView* определяет цвет линии и границ отображаемой геометрической фигуры соответственно.

FrameStyle – тип линий – строка. Возможные значения: *solid*, *dash*, *dot*, *dashdot*, *dashdotdot*, *double*. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений. Для элементов типа *TextView*, *ImageView*, *BarcodeView* и *ChartView* определяет тип линий рамки вокруг элемента, определенных в свойстве *FrameEdges*. Для элементов типа *LineView* и *ShapeView* определяет тип линий отображаемой геометрической фигуры;

FrameWidth – толщина линий – вещественное. Для элементов типа *TextView*, *ImageView*, *BarcodeView* и *ChartView* определяет толщину линий рамки вокруг элемента, определенных в свойстве *FrameEdges*. Для элементов типа *LineView* и *ShapeView* определяет толщину линий отображаемой геометрической фигуры;

ZIndex – порядок перекрытия элемента – целое. Определяет порядок перекрытия одних элементов другими.

Кроме общих атрибутов каждый тип *элементов отображения данных* имеет свой уникальный набор атрибутов:

TextView

Align – выравнивание содержимого внутри элемента по горизонтали – строка. Возможные значения: *left*, *center*, *right*, *justify*. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений;

VAlign – выравнивание содержимого внутри элемента по вертикали – строка. Возможные значения: *top*, *middle*, *bottom*. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений;

LetterSpacing – расстояние между символами в строке по горизонтали – вещественное;

LineHeight – высота строки – вещественное. Определяет расстояние между строками текста по вертикали при многострочном отображении текста;

DisplayFormat – формат отображения\шаблон форматирования данных – строка. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности шаблонов формата отображения;

FontName – имя шрифта – строка;

FontColor – цвет шрифта – строка. HEX-представление цвета RGB;

FontSize – размер шрифта – вещественное;

FontStyle – битовое представление стиля начертания шрифта в десятичном виде – целое. Первый бит – жирный шрифт, второй бит – наклонный шрифт, третий бит – зачеркнутый шрифт;

FrameEdges – битовое представление отображаемых рамок вокруг границ элемента в десятичном виде – целое. Первый бит – левая граница; второй – верхняя; третий – нижняя; четвертый – правая. Таким образом десятичное значение 15 (двоичное 1111) означает отображение всех четырех границ вокруг элемента, а десятичный 0 (двоичное 0000) означает, что ни одна граница отображена не будет. На Рисунке 3 показана схема битового представления отображаемых рамок. Разработчик модуля генерации отчетов и конструктора шаблонов позаботиться о согласовании последовательности бит в этом двоичном представлении;

Value – значение, отображаемое внутри элемента – строка. В качестве значения этот атрибут может принимать как напрямую введенные константные значения (текст, числа, даты), так и через *плейсхолдеры* (*placeholder*): имена переменных, параметров, полей dataset'ов, имена функций, системных переменных и т.д., а также всевозможные комбинации из перечисленного. Способ передачи данных в *элементы отображения данных* через *плейсхолдеры*, а также их типы будут описаны ниже в разделе «*Компиляция шаблона*»;

Visible – определяет, будет ли данный *TextView* видим в экспортируемом отчете – логическое. Возможность скрывать некоторые поля может быть очень полезна для организации вычисляемых полей и сложных расчетов при компиляции шаблона отчета.

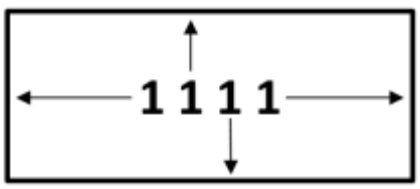

Рисунок 3. «Битовое представление отображаемых рамок вокруг границ элемента»

ImageView

Path – путь к файлу с изображением – строка;

Proportional – использовать пропорциональное масштабирование – логическое;

Center – располагать изображение в центре при масштабировании – логическое;

Stretch – растягивать маленькие и сжимать большие изображения под размер элемента на странице – логическое.

BarcodeView

Type – тип штрих-кода, который необходимо вывести – строка. Список возможных значений ограничивается списком существующих\необходимых типов штрих-кодов. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений;

Value – значение, которое будет отображено в виде штрих-кода – строка. Функционально аналогичен одноименному атрибуту для элемента *TextView* – возможна передача данных как в виде константных значений, так и через *плейсхолдеры*;

FillColor – цвет отображаемого штрих-кода – строка. HEX-представление цвета RGB.

ShapeView

Type – тип отображаемой геометрической фигуры – строка. Возможные значения: *ellipse*, *rectangle*, *triangle*, *diamond*. Разработчик компилятора и конструктора шаблона для генератора отчета должны позаботиться о согласованности имен возможных значений;

FillColor – цвет заливки отображаемой геометрической фигуры – строка. HEX-представление цвета RGB. (*none* – без заливки цветом, прозрачный фон).

Описание структуры элемента ***ChartView*** не входит в материал настоящей статьи из-за ограничений по объему. Предлагаю читателю самостоятельно изучить структуру параметров и принципы построения диаграмм и графиков на примерах готовых решений – например, известный всем и хорошо документированный Chart.js [4].

Теперь, когда мы разобрались с типом ***FieldSet*** и его структурными элементами, восстановим последовательность повествования и вернемся к описанию следующего элемента в структуре шаблона отчета – ***Chapters***.

Chapters – массив элементов шаблона, отвечающих за описание *разделов* (***Chapter***) отчета. Данный элемент, по сути, является коллекцией самостоятельных шаблонов отчетов, каждый из которых может иметь уникальную внутреннюю структуру: собственный уникальный дизайн отчета и свои независимые наборы данных. При этом доступ каждого такого *раздела* к глобальным объектам отчета сохраняется: поля элементов ***ReportVariables***, ***InputParameters***, ***ReportDataset***, описанные во внешней части структуры шаблона, также будут доступны для использования во внутренних частях шаблона, независимо от того, как глубоко они расположены в иерархической структуре шаблона. Каждый шаблон отчета должен содержать хотя бы один *раздел* – разработчику конструктора шаблонов отчетов необходимо учесть этот факт.

Для общего понимания способов использования разделов давайте рассмотрим один из вариантов использования. На Рисунке 4 представлен пример структуры шаблона отчета с пятью *разделами*: первый *раздел* использован в качестве титульного листа для всего отчета, далее идут три *раздела* с данными, являющимися телом отчета, и замыкает список *разделов* резюме отчета – последняя страница.

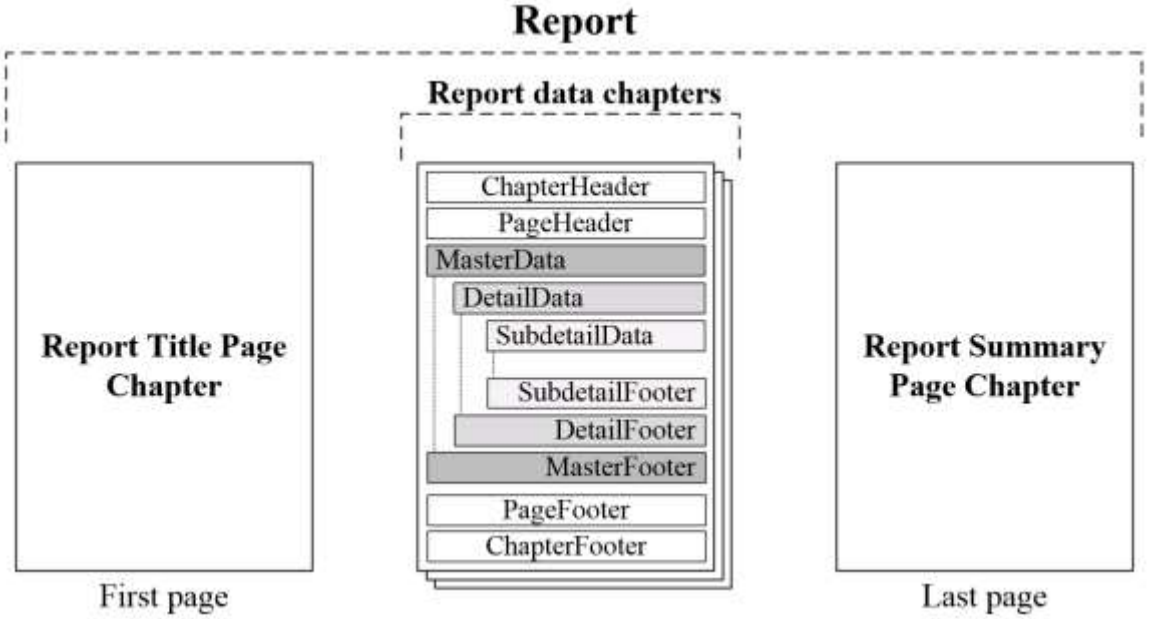

Рисунок 4. «Пример структуры шаблона отчета»

Таким образом, комбинируя разделы с разным дизайном и структурой, можно строить отчеты под самые разные нужды пользователей Информационных систем.

Итак, давайте разберем структуру элемента ***Chapter***:

PageParameters – аналогично элементу *PageParameters* корневого элемента *Report*, определяет параметры страницы для текущего *раздела* отчета. Необязательный элемент. При отсутствии элемента *PageParameters* у *раздела* следует применять параметры страницы, определенные в элементе *PageParameters* корневого элемента *Report*. Разработчику модуля экспорта для генератора отчетов следует учесть эту особенность;

ChapterHeader – контейнер заголовка раздела отчета – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится один раз только на первой странице раздела. Имеет единственный атрибут *Height* – высота контейнера – вещественное;

PageHeader – контейнер заголовка страницы раздела – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится вверху каждой страницы внутри текущего раздела отчета. Аналог верхнего колонтитула в Microsoft Word и подобных редакторах. Имеет единственный атрибут *Height* – высота контейнера – вещественное;

DataSection – секция данных. Обязательный элемент. Является контейнером для элементов типа *FieldSet*, определяющих заголовок данных, данные и подвал данных (см. Рисунок 1 блоки *MasterHeader*, *MasterData* и *MasterFooter*). При необходимости построения многоуровневых отчетов секция данных может содержать вложенный *DataSection* – глубина вложенности теоретически не ограничена. Имеет необязательный атрибут *DataSet* – текст запроса к базе данных, результат которого будет выводиться в этой секции отчета. Способ передачи параметров в запрос к базе данных будет описан ниже в разделе «*Компиляция шаблона*». Внутреннее строение секции данных состоит из:

Header – контейнер заголовка секции данных – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится один раз в начале секции (см. Рисунок 1 блок *MasterHeader*). Имеет единственный атрибут *Height* – высота контейнера – вещественное;

Data – контейнер данных – тип *FieldSet*. Обязательный элемент, если в секции данных определен *DataSet*. При экспорте отчета выводится под контейнером *Header* столько раз, сколько записей возвратил запрос *DataSet* текущей секции данных (см. Рисунок 1 блок *MasterData*). Имеет единственный атрибут *Height* – высота контейнера – вещественное;

Footer – контейнер для подвала секции данных – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится под контейнером *Data* один раз в конце секции (см. Рисунок 1 блок *MasterFooter*). Предназначен в основном для вывода итогов по данным секции: суммы, количество строк и т.д. Имеет единственный атрибут *Height* – высота контейнера – вещественное;

DataSection – вложенная секция данных. Необязательный элемент.

PageFooter - контейнер подвала страницы раздела – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится внизу каждой страницы внутри текущего раздела отчета. Аналог нижнего колонтитула в

Microsoft Word и подобных редакторах. Имеет единственный атрибут *Height* – высота контейнера – вещественное;

ChapterSummary - контейнер подвала раздела отчета – тип *FieldSet*. Необязательный элемент. При экспорте отчета выводится один раз только на последней странице раздела. Имеет единственный атрибут *Height* – высота контейнера – вещественное;

Помимо этого, элемент ***Chapter*** должен иметь атрибут ***PrintOrder***, который определяет очередность компиляции и экспорта *раздела*: в отчетах с несколькими разделами разработчик шаблона отчета должен определять, какой раздел будет обрабатываться первым, какой вторым и т.д. Разработчику компилятора и модуля экспорта для генератора отчетов следует учесть эту особенность.

На этом закончим описание формата шаблона отчета и далее перейдем к описанию принципов работы генератора отчетов. А сейчас кратко остановимся на конструкторе шаблонов.

Конструктор шаблонов отчетов

Для удобства и быстроты построения шаблонов отчетов для Ваших Информационных систем Вам потребуется реализовать программный инструмент – Конструктор шаблонов. Этот конструктор должен представлять собой визуальный редактор, где разработчик шаблона отчета будет выполнять свою работу. Функционально Конструктор шаблонов будет схож с любой современной IDE, где есть функционал разработки пользовательских интерфейсов (Microsoft Visual Studio, Delphi, C++ Builder, XCode и т.п.). В таких средах разработки существует палитра компонентов, которую разработчик может вставлять на рабочее поле, и есть редактор атрибутов для выбранного на рабочем поле компонента. Изучите, если ранее не сталкивались, принцип построения пользовательских интерфейсов в указанных (или подобных им) IDE. Этот принцип во многом схож с принципом построения шаблона отчета в нашем Конструкторе – аналогичным образом Вам потребуется реализовать работу с элементами шаблона и их атрибутами. Разработчику конструктора шаблонов отчетов следует реализовать функцию предварительного просмотра для проектируемого шаблона – это многократно повысит эффективность труда разработчика шаблонов. Рекомендую выполнять реализацию Конструктора шаблонов в том-же web-стеке, что и Ваша Информационная система – когда Ваш заказчик пожелает иметь встроенный Конструктор шаблонов отчетов в своей Информационной системе, Вы легко сможете его туда интегрировать.

Процесс генерации отчета

Как упоминалось выше, процесс генерации отчета состоит из двух этапов: *компиляция шаблона* и *экспорт отчета*. Выполнением этих этапов в Вашей Информационной системе будет заниматься Генератор отчетов – серверный модуль, запускаемый основным сервисом Вашей Информационной системы. Подробнее об этом я расскажу в разделе «***Интеграция генератора отчетов в Информационную систему***».

Компиляция шаблона

Компиляцией шаблона называется процесс, во время которого компилятор генератора отчетов, применяя алгоритм рекурсивного обхода древовидной структуры [3] шаблона, последовательно выполняет следующие процедуры:

Подготовку запросов к базе данных, указанных в атрибутах DataSet – замена *плейсхолдеров* в запросах на значения входных параметров и\или значений переменных, констант, значений функций и т.д.;

Подключение к базе данных, выполнение подготовленных запросов и построчное заполнение контейнеров *Data* в секциях данных *DataSection* шаблона отчета полученными данными;

Рекурсивную замену *плейсхолдеров* в атрибутах Value у элементов отображения данных на соответствующие значения;

В контексте этой статьи *плейсхолдерами* я называю особым образом оформленные строковые метки-идентификаторы, которые заменяются компилятором в процессе компиляции шаблона на значения, соответствующие типу и имени этого *плейсхолдера*. Тип *плейсхолдера* определяется его первым символом, а имя – следующая за первым символом строка-идентификатор. В качестве имени *плейсхолдера* допустимо использовать строку, состоящую из букв латинского алфавита и цифр. Недопустимо, чтобы имя *плейсхолдера* начиналось с цифры. В качестве типа *плейсхолдера* допустимо использовать печатные символы, перечисленные в Таблице 1.

Таблица 1.

Определения типов и назначения плейсхолдеров

Символ	Назначение	Пример написания
@	Входные параметры отчета	@beginDate, @endDate
#	Поля данных *ReportDataset*	#fieldName
$	Переменные *ReportVariables*	$myVar9
&	Поля данных *Dataset* текущей секции данных	&id, &name, &cost
_	Значения элементов *TextView*	_TextView32
!	Встроенные переменные и константы генератора отчетов	!date, !time, !pageNo,

		Report date: !date. Pages total: !pagesCount.
=	Встроенные функции	=Sum(_TextView11)
^(…)	Арифметические операции для калькулируемого поля	^(_TextView1+_TextView2) ^(&cost / 100 * $myVar9)
|	Присвоить отображаемое или вычисленное значение указанной переменной	=Sum(_TextView11) | $myVar9 ^(_TextView1+_TextView2) | $m ^(&cost + $total) | $total

Показанный список в достаточной степени покрывает базовые потребности разработчика шаблонов отчетов, но Вы можете добавить необходимые в Вашем генераторе отчетов типы *плейсхолдеров*. Как говорилось ранее и как видно в колонке «Пример написания» Таблицы 1, *плейсхолдеры* можно комбинировать с обычным текстом и между собой в любых комбинациях – для этого процедура компилятора, выполняющая замену *плейсхолдера* на его значение, должна быть рекурсивной – обработка комплексных *плейсхолдеров* требует этого. Разработчику компилятора для генератора отчетов следует учесть этот факт.

Из представленного списка *плейсхолдеров* наибольший интерес представляют два из них:

Встроенные переменные и константы генератора отчетов. Разработчику компилятора шаблона и модуля экспорта необходимо реализовать такие важные переменные генератора как: дата\время компиляции отчета, количество строк в текущей секции данных, номер строки в текущей секции данных, номер текущей страницы в разделе (в отчете), общее количество страниц в разделе (в отчете) и т.д. Важно отметить, что не все из этих переменных могут быть заменены на соответствующие значения при компиляции шаблона: значения переменных, относящиеся к нумерации страниц, например, можно получить только на этапе экспорта отчета в формат, подразумевающий разбивку данных отчета на страницы. Таким образом, разработчикам генератора отчетов следует внимательно подойти к реализации работы со встроенными переменными генератора отчетов – какие из них должны быть реализованы в компиляторе шаблонов, а какие в модуле экспорта.

Встроенные функции. Реализуйте в компиляторе шаблона хотя бы набор базовых функций: Сумма, Среднее, Медиана, Максимум, Минимум, Тригонометрические и логические функции. Полное разнообразие функций, которое может пригодиться при построении отчетности, Вы сможете посмотреть, например, в программе Microsoft Excel и выбрать необходимые для реализации в своем генераторе отчетов.

Результатом работы компилятора является файл *скомпилированного отчета*. Структурно этот файл схож с шаблоном отчета, но уже заполнен данными: *плейсхолдеры* заменены соответствующими значениями, а секции данных заполнены результатами выполнения запросов к базе данных, которые были указаны в атрибутах ***DataSet***. Именно из этого файла модуль экспорта отчета будет формировать выходной файл отчета.

Экспорт отчета

Процессом экспорта отчета называется процесс преобразования скомпилированного отчета в формат, выбранный пользователем Информационной системы при создании отчета. Среди выходных форматов отчетов наиболее распространенными являются: PDF, Excel, Word и HTML. Вы можете дополнить этот перечень необходимыми форматами файлов. При реализации модуля экспорта отчетов удобно будет предусмотреть систему подключаемых плагинов – отдельный плагин для каждого типа выходного формата. Подобно компилятору шаблона модуль экспорта выполняет рекурсивный проход по структуре файла скомпилированного отчета и выполняет перенос данных из его элементов в выходной файл в соответствии с правилами, определенными в атрибутах этих элементов: позиция и размеры элемента; тип, размер, цвет и стиль шрифта; границы и заливка элемента и т.д.

На рисунке 5 представлен вид готового отчета, прошедшего этап компиляции шаблона и этап экспорта в формат PDF.

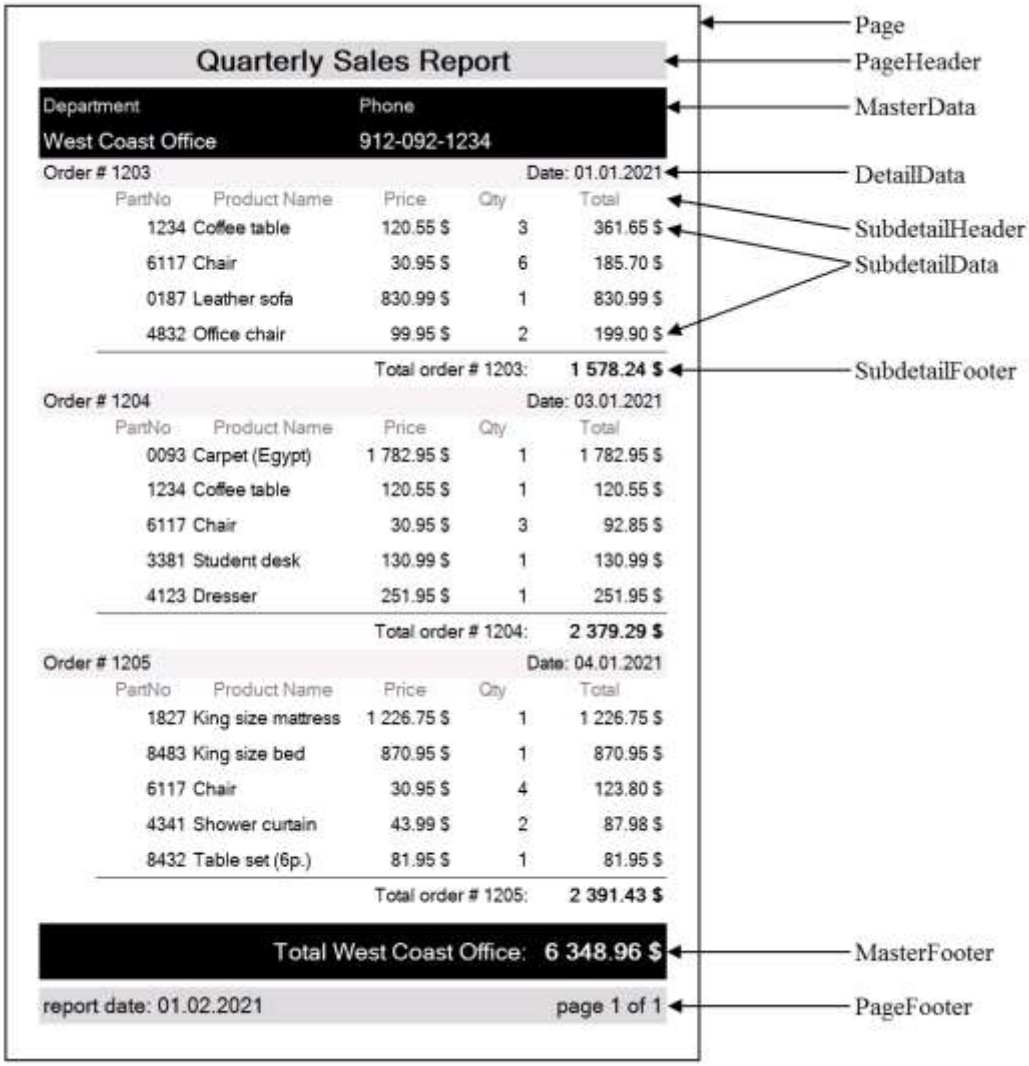

Рисунок 5. «Вид готового отчета»

Интеграция генератора отчетов в Информационную систему

Для интеграции модуля Генератора отчетов в Вашу Информационную систему Вам понадобится:

Добавить необходимые объекты в базу данных:

Таблица для хранения списка отчетов, установленных и доступных для использования в Информационной системе;

Таблица для распределения доступа пользователей Информационной системы к отчетам;

Таблица для хранения информации об отчетах пользователей: хранит информацию о том, какой пользователь, когда и какой отчет формировал, какие входные параметры передавал, статус готовности отчета и пр.

Доработать основой сервис – добавить методы в API для работы с отчетами:

Отдача списка отчетов, доступных текущему пользователю;

Отдача списка входных параметров для выбранного отчета (см. *InputParameters* в разделе *Структура шаблона* выше);

Получение данных запроса на создание нового отчета (id пользователя, id выбранного отчета, входные параметры отчета, тип выходного формата отчета), сохранение данных в Таблицу отчетов пользователя и запуск модуля Генератора отчетов;

Отдача файла готового отчета.

Доработать frontend Информационной системы – Добавить пункт главного меню «Мои отчеты», перейдя в который пользователь попадет на страницу своих отчетов, где:

Панель с кнопками «Создать новый отчет» и «Удалить отчет»;

При нажатии на кнопку «Создать новый отчет» пользователь должен получать модальную форму, на которой сначала выбирает из выпадающего списка необходимый ему тип отчета. При выборе отчета должен отобразиться соответствующий этому типу отчета список полей для ввода входных параметров отчета и выпадающий список для выбора формата выходного файла;

Ниже кнопочной панели – Список отчетов в табличном виде, сформированных ранее текущим пользователем, где следует, помимо основных данных о параметрах отчета, отобразить кнопку для скачивания готового выходного файла отчета;

Хотелось бы пояснить касательно процесса запуска модуля Генератора отчетов, упомянутого в этом списке требований: Основной сервис Вашей информационной системы, при запуске модуля Генератора отчетов, должен передать ему идентификатор сохраненной записи в таблице отчетов пользователя как параметр командной строки. На рисунке 6 показана схема взаимодействия Информационной системы и модуля Генератора отчетов: пунктирная стрелка от Основного сервиса ***Main Service Backend*** к модулю Генератора отчетов ***Report Generator*** показывает процесс его запуска. ***Report Generator*** при старте считывает из собственного файла конфигурации параметры подключения к базе данных Информационной системы ***Database*** и получает все параметры для построения отчета из таблицы отчетов пользователя по полученному в параметре командной строки идентификатору этого отчета. В этих параметрах, среди прочего, содержится имя шаблона, файл которого хранится в каталоге ***Template Storage***.

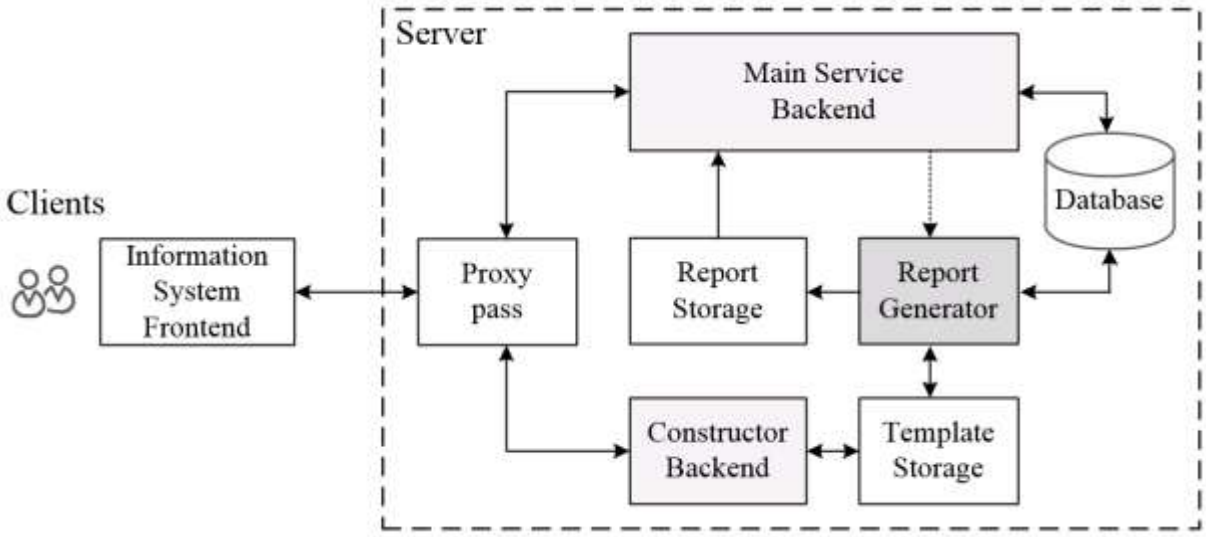

Рисунок 6. «Принципиальная схема взаимодействия Информационной системы с Генератором отчетов»

В процессе своей работы модуль Генератора отчета меняет значение статуса готовности отчета в таблице отчетов пользователя. Результатом работы ***Report Generator*** являются файлы скомпилированного и экспортированного отчета, которые сохраняются в каталоге ***Report Storage*** – именно из этого каталога ***Main Service Backend*** отдает файлы готовых отчетов пользователям, когда они выполняют запрос готового отчета.

Предложенная схема подразумевает асинхронное взаимодействия модулей. Преимуществами такого метода взаимодействия модулей являются:

Система, построенная по предложенной схеме, соответствует требованиям микросервисной архитектуры [2] и легко может быть распределена внутри серверной инфраструктуры;

Пользователь, запустивший процесс формирования отчета, может не дожидаться завершения этого процесса и заниматься другой работой, а за результатом работы Генератора отчетов обратиться в раздел «Мои отчеты» в любое другое время;

Процесс формирования отчетности в информационной системе может выполняться без участия пользователя – по расписанию.

Список усовершенствований.

Напоследок позвольте представить Вашему вниманию несколько рекомендаций по улучшению функциональности Вашей подсистемы формирования отчетности:

Добавьте возможность определения толщины и типа линий для рамки вокруг элементов отображения данных отдельно для каждой из сторон рамки;

Добавьте условное форматирование для *TextView* – в зависимости от значения, отображаемого внутри элемента, применять то или иное форматирование: цвет фона, параметры рамки, стиль\цвет шрифта. (например: все значения > 5 отображать жирным шрифтом);

Будет полезно добавить возможность вывода контейнера заголовка секции данных (*Header*) вверху каждой страницы, на которую выводятся строки этого контейнера данных (*Data*);

Добавьте планирование отчетов – формирование и рассылка отчетов по расписанию;

Добавьте возможность отправки сформированного отчета другим пользователям Информационной системы – по почте или в раздел «Мои отчеты». Пользователь должен сам выбрать, каким пользователям отправить отчет;

Защитите свою Информационную систему и шаблоны отчетов от несанкционированных изменений [1].

Список литературы:

1. Плетнев А.В. Защита проприетарного программного продукта от несанкционированных изменений // Universum: технические науки: электрон. научн. журн. 2021. 9(90). – URL: https://7universum.com/ru/tech/archive/item/12268 (дата обращения: 20.11.2021).
2. Ньюмен Сэм. Создание микросервисов (с. 126). – СПб.: ИД «Питер», 2016.
3. Ахо Альфред В., Ульман Джеффри Д., Хопкрофт В. Структуры данных и алгоритмы. – М.: ИД «Вильямс», 2016.
4. Chart.js – электронная документация. URL: https://www.chartjs.org/docs/latest/ (дата обращения: 21.11.2021)
5. XML – онлайн энциклопедия. URL: https://ru.wikipedia.org/wiki/XML (дата обращения: 24.11.2021).

References:

1. Pletnev A.V. Protection of proprietary software product from unauthorized changes // Universum: tehnicheskie nauki: scientific Internet-journal. 2021. 9(90). – URL: https://7universum.com/ru/tech/archive/item/12268 (date of access: 20.11.2021). (In Russian).
2. Newman Sam. Building Microservices (p. 126). – St.P.: «Piter», 2016. (In Russian).
3. Aho Alfred, Ullman Jeffrey, Hopkroft John. Data structures and algorithms. – M.: «Williams», 2016. (In Russian).
4. Chart.js – online documentation. URL: https://www.chartjs.org/docs/latest/ (date of access: 21.11.2021). (In Russian).
5. XML – online encyclopedia. URL: https://ru.wikipedia.org/wiki/XML (date of access: 24.11.2021). (In Russian).