Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

# ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Модуль интеграции с системой бизнес-аналитики**

Студент гр. 43501/3 П.В. Супронович

Санкт-Петербург – 2017

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Работа допущена к защите
зав. кафедрой

_____ В.М. Ицыксон

«____» _____ 2017 г.

# ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Модуль интеграции с системой бизнес-аналитики**

Направление: 09.03.01 – Информатика и вычислительная техника

Выполнил студент гр. 43501/3 _____ П.В. Супронович

Научный руководитель,
ст. преподаватель _____ А.В. Зозуля

Санкт-Петербург – 2017

# РЕФЕРАТ

Отчет, 36 стр., 14 рис., 4 табл., 10 ист., 1 прил.

ИНФОРМАЦИОННЫЕ СИСТЕМЫ, БИЗНЕС-АНАЛИТИКА, PENTAHO, VTIGER, ИНТЕГРАЦИЯ.

В выпускной работе проводится разработка модуля интеграции системы бизнес-аналитики Pentaho BI с CRM-системой Vtiger для расширения возможностей аналитики данных Vtiger.

Был проведён анализ инструментов построения отчётов, в результате чего был выбран конструктор аналитических отчётов Saiku analytics.

Средством разработки является объектно-ориентированный язык программирования Java. Взаимодействие между CRM-системой и модулем интеграции для платформы Pentaho реализовано с помощью REST-интерфейса.

В результате работы получен полноценный модуль Pentaho с графическим интерфейсом, предоставляющий аналитические отчёты и позволяющий генерировать REST-запросы для разработанного модуля.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Необходимость улучшить обслуживание клиентов и автоматизировать взаимодействие с ними приводит всё больше компаний к решению использовать специальные информационные системамы.

Одним из таких решений является CRM(Customer relationship management) - термин, характеризующий практики, стратегии и технологии, используемые компаниями для управления и анализа данных и взаимодействий с пользователем.[1]

Основные причины внедрения системы управления взаимоотношениями:

- Ведение общей клиентской базы.
- Повышение эффективности продаж;
- Повышение эффективности маркетинга;
- Повышение уровня обслуживания клиентов;
- Отчетность и анализ деятельности отделов, связанных со взаимодействием с клиентами.

К сожалению, в большинстве таких систем возможности аналитики данных ограничиваются формированием отчётности только по вопросам отношений с клиентами (сколько привлечено клиентов, прогнозы по продажам и т.д.), чего во многих случаях не хватает для бизнес-анализа.

Для решения данной проблемы многие организации используют более мощные средства и платформы - системы бизнес-аналитики. Business Intelligence (BI) - это набор методов и инструментов для объединения, обработки и предоставления информации в удобной, осмысленной форме.

Однако, возникает проблема взаимодействия данных систем друг с другом. Наилучшим решением является интеграция CRM-системы с BI-платформой.

Целью данной работы является разработка модуля интеграции аналитических отчётов платформы Pentaho BI в CRM-систему Vtiger. Данный модуль позволит экспортировать данные из отчётов, находящихся в хранилище BI-платформы для дальнейшей работы с ними в CRM-системе.

# 1. Анализ существующих решений

## 1.1. Информационные системы

Информационные системы - это взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации для достижения цели управления.

### 1.1.1. Способы аналитики информационных систем

В информационных системах доступны следующие способы анализа и визуального представления результатов:

1. Инструменты для анализа данных. Это специальные программы, дающие возможность создания отчётов, диаграмм и срезов данных. Особенностью таких отчётов является простота использования за счёт графического интерфейса. Такие отчёты могут выполнять непосредственно работники, использующие систему.

2. Программные отчёты. Такие отчёты используются в случаях, когда функциональности мастеров отчётов недостаточно. Программные отчёты требуют более широкого знания системы и навыков написания на языке SQL. Обычно такой способ - это трудоёмкий ручной процесс, требующий специальных навыков.

### 1.1.2. Инструменты аналитики Vtiger

CRM система – это программное обеспечение, предназначенное для сбора и хранения всей информации, связанной с клиентами, а также автоматизации взаимодействий с ними. Целю CRM является предоставление полного набора данных о клиентах, который может использоваться для увеличения продаж, удержания, привлечения и повышения уровня обслуживания клиентов.

Vtiger CRM - это система управления взаимоотношениями с клиентами с открытым кодом и возможностью создания пользовательских модулей. Vtiger CRM предоставляет сделующие средства для анализа и интеллектуальной обработки данных: отчёты, контрольные панели и виджеты.

"**Виджет** - это небольшой графический элемент, выводимый на главную страницу системы Vtiger CRM, либо на страницу “Виджеты” для любых Модулей системы."[2]

Виджет, как можно увидеть на рисунке 1.1, визуализирует представление данных в системе. На одном экране можно поместить сразу несколько виджетов.

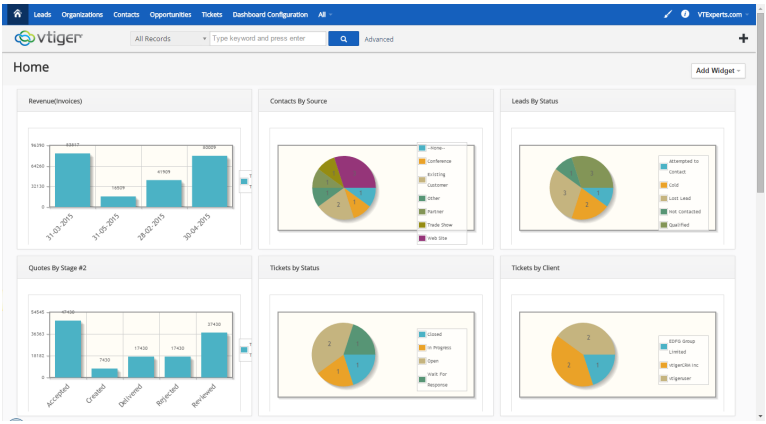


Рисунок 1.1. Пример виджетов Vtiger

Другим средством является **Модуль Отчёты**, который собирает, фильтрует и организовывает данные в виде таблиц и диаграмм. Системой Vtiger CRM предоставляется генератор и дизайнер отчётов:

- Генератор отчетов создает отчеты, которые можно просматривать и выводить в файлы PDF или Excel.
- Дизайнер отчетов даёт возможность выбирать данные, которые необходимо включить в отчет, и создать представление данных в отчете.

Как видно на рисунке 1.2, Vtiger предлагает множество вариантов составления стандартных отчётов.

Существует два вида отчётов: **диаграммы и детальные отчёты**. Детальные отчёты дают возможность пользователю получить релевантные для него данные в табличной форме. Пример такого отчёта на рисунке 1.3.

| Название Отчета | Описание | Имя Папки |
|---|---|---|
| Контакты по Контрагентам | Контакты, относящиеся к Контрагентам | Отчеты по Контактам и Контрагентам |
| Контакты без Контрагентов | Контакты, не относящиеся к Контрагентам | Отчеты по Контактам и Контрагентам |
| Контакты по Сделкам | Контакты относящиеся к Сделкам | Отчеты по Контактам и Контрагентам |
| Обращения по Источникам | Обращения по Источникам | Отчеты по Обращениям |
| Отчет по Статусу Обращения | Отчет по Статусу Обращения | Отчеты по Обращениям |
| Конвейер Сделок | Конвейер Сделок | Отчеты по Сделкам |
| Закрытые Сделки | Сделки с Выигрышем | Отчеты по Сделкам |
| Деятельность за прошлый месяц | Деятельность за прошлый месяц | Отчеты по Деятельности |
| Деятельность за этот месяц | Деятельность за этот месяц | Отчеты по Деятельности |
| Заявки по Товарам | Заявки, относящиеся к Товарам | Отчеты по Заявкам в Поддержку |
| Заявки по Приоритетам | Заявки по Приоритетам | Отчеты по Заявкам в Поддержку |
| Открытые Заявки | Открытые Заявки | Отчеты по Заявкам в Поддержку |
| Информация о Товаре | Детальный отчет по Товарам | Отчеты по Товарам |
| Товары по Контактам | Товары относящиеся к Контактам | Отчеты по Товарам |

Рисунок 1.2. Список стандартных отчётов

Диаграмма в свою очередь - это визуальное представление данных в виде круговой или столбиковой диаграммы, или линейного графика. Пример диаграммы на рисунке 1.4.

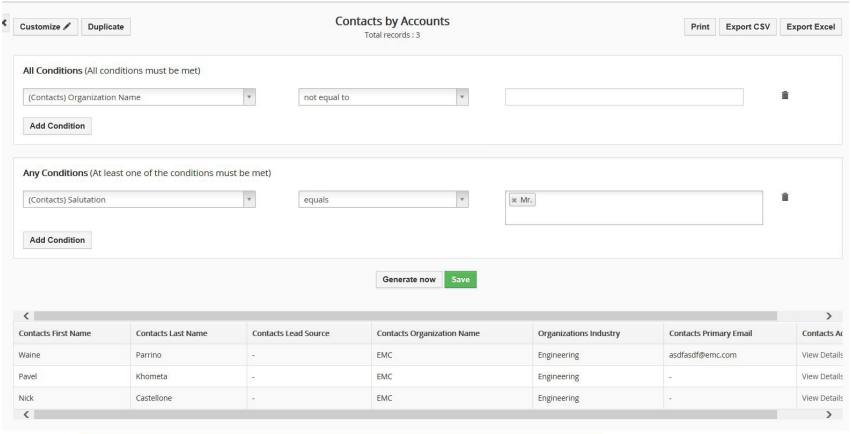


Рисунок 1.3. Пример детального отчёта

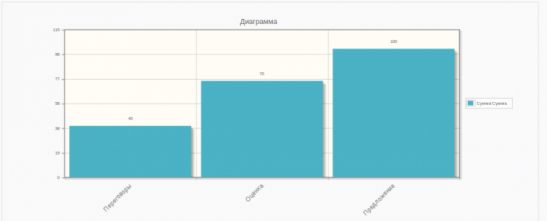


Рисунок 1.4. Пример отчёта-диаграммы

Обоим видам отчётов необходима дополнительная настройка с использованим фильтров. Однако они не всегда являются полными для аналитики, так как относятся только к сфере продаж и маркетинга. Также, в таких отчётах отсутствует гибкость.

## 1.2. Системы бизнес-аналитики

Business Intellegence (BI) системы - это аналитические системы, которые объединяют данные из различных источников информации, обрабатывают их и предоставляют удобный интерфейс для изучения и оценки полученных сведений.[4] Структура BI платформы представлена на рисунке 1.5.

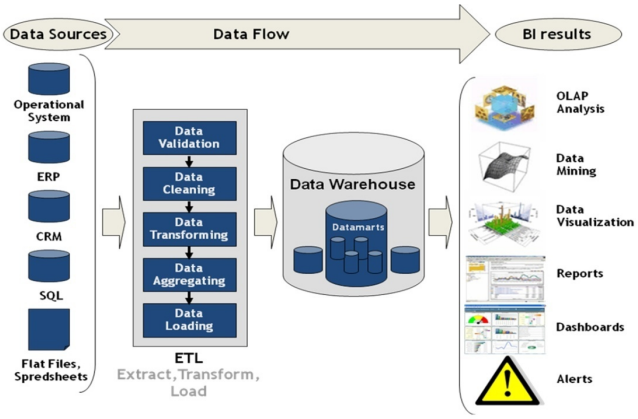


Рисунок 1.5. Структура BI платформы

### 1.2.1. Анализ данных платформы Pentaho BI

Pentaho BI - свободно распространяющаяся система бизнес-аналитики. Сервер платформы написан на языке Java и предоставляет инфраструктуру для отображения отчётов через веб-сервисы.

В платформе Pentaho есть два основных варианта анализа данных: Pentaho Reporting и Mondrian OLAP Server.

1. Pentaho Reporting - это набор инструментов для создания детальных отчётов. [5]

   С помощью Pentaho Reporting можно преобразовывать данные в приспособленную для клиентов значимую информацию. В Pentaho Reporting возможно создавать отчеты в форматах HTML, Excel, PDF, Text, XML, CSV, или печатной форме.

   Также одним из преимуществ является отсутствие требований к знаниям о системе и простота в использовании (рисунок 1.6).

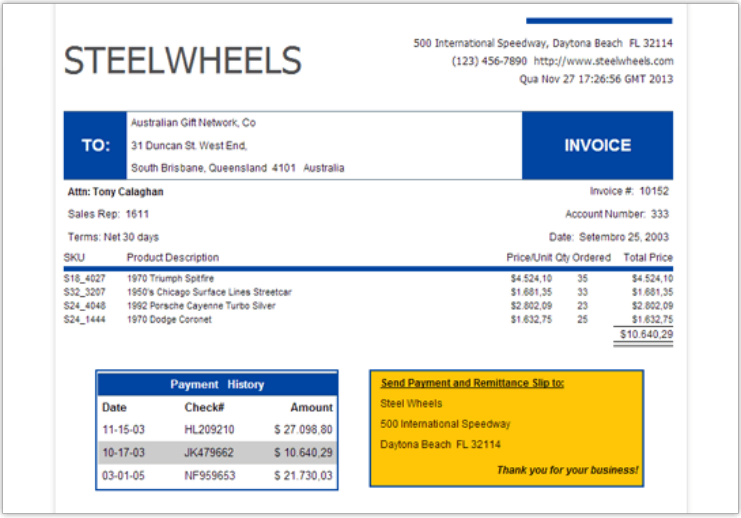


Рисунок 1.6. Отчёт в Pentaho Reporting

2. Mondrian OLAP Server позволяет пользователям анализировать сразу большие массивы данных, строить прогнозы и сценарии,

а также разрабатывать множество вариантов планов за счёт OLAP-технологии многомерных кубов. При построении аналитического отчёта пользователь делает срезы измерений гиперкуба, чтобы изолировать и найти необходимые данные. Это даёт гибкость предоставления анализируемых данных. Также пользователю не обязательно знать языки выборки, и необходимы минимальные знания системы. Пример аналитического отчёта приведён на рисунке 1.7.

| | USA | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CA | | | | OR | | WA | | | | | | |
| | Beverly Hills | Los Angeles | San Diego | San Francisco | Portland | Salem | Bellingham | Bremerton | Seattle | Spokane | Tacoma | Walla Walla | Yakima |
| Product Family | Store 6 | Store 7 | Store 24 | Store 14 | Store 11 | Store 13 | Store 2 | Store 3 | Store 15 | Store 16 | Store 17 | Store 22 | Store 23 |
| Drink | 590,419 | 850,181 | 746,353 | 105,698 | 324,723 | 1 732,508 | 141,182 | 777,301 | 990,566 | 665,289 | 1 822,451 | 34,357 | 437,315 |
| Food | 3 635,132 | 8 725,71 | 8 905,011 | 662,178 | 2 696,758 | 11 030,067 | 696,042 | 8 760,093 | 8 596,451 | 3 901,889 | 11 113,203 | 983,866 | 3 661,138 |
| Non-Consumable | 907,346 | 2 059,027 | 2 199,299 | 220,944 | 1 021,476 | 3 368,456 | 212,235 | 2 122,231 | 1 930,098 | 1 214,785 | 3 165,165 | 75,546 | 1 175,984 |

Рисунок 1.7. Отчёт, построенный инструментом Saiku

Сравнение представлено в таблице 1.1:

| Reporting | Analysis |
|---|---|
| Преобразует данные в информацию | Преобразует данные и информацию в ответы |
| Вызывает вопросы о бизнесе у своих конечных пользователей | Отвечает на вопросы, интерпретируя данные на более глубоком уровне и предоставляя действенные рекомендации |
| Показывает, что произошло | Показывает, почему это произошло и что с этим можно сделать |

Таблица 1.1. Сравнение Reporting и Analysis

За отчётностью идёт аналитика, чтобы ответить на появившиеся вопросы.

### Конструкторы аналитических отчётов

Pentaho BI обладает несколькими приложениями для построения аналитических отчётов:

1. **Jpivot** - это библиотека пользовательских тегов JSP, которая отображает таблицу OLAP и позволяет пользователям выполнять типичные OLAP-навигации. Поддержка данного инструмента разработчиками завершилась в 2008 году, из-за чего он, на данный момент, является наименее удобным и бытрым вариантом построения аналитических отчётов.

2. **Saiku** является приложением OLAP-анализа. Обладает легковесной архитектурой клиента, а также простой реализацией REST-API. Скорость работы данного инструмента выше, чем у остальных представленных конструкторов аналитических отчётов.

3. **Pivot4j** - Java API для OLAP-серверов. Предоставляет широкие возможности для реализации независимого API. Pivot4j практически все операции выполняет на стороне сервера. Однако, реализация REST-приложения сложнее, чем у Saiku.

## 1.3. Интеграция BI и информационной системы

Хотя, и BI и CRM решения относятся к сбору и анализу данных, однако, если BI, предоставляет пользователю принимать решения, то CRM, напротив, позволяет пользователям участвовать в любом процессе анализа данных и автоматизирует их.

Очевидно, что связь приложения CRM с системой бизнес-аналитики даёт преимущества обеих систем(эффективный анализ со стороны BI и улучшение и автоматизация взаимодействия с клиентами и увеличение клиентской базы). BI обрабатывает большие объёмы информации, а CRM-система выступает провайдером данных. Однако, такая связь обычно приводит также к дополнительным затратам на приобритение программных средств.

Кроме того, для добавления пользовательских модулей и построения нестандартных отчётов в CRM-системе необходимо изучить язык программирования, на котором написано CRM-приложение. Использование бизнес-аналитики позволит сократить необходимость дополнительных знаний в CRM и сосредоточиться непосредственно на анализе данных.

## 1.4. Итоги

Были рассмотрены инструменты анализа данных CRM-системы Vtiger. В результате выявлено, что несмотря на приличный набор инструментов, они не отличаются гибкостью и их недостаточно для полноценного анализа данных.

Более того, создание отчётов в CRM требует привлечения специалиста и чаще всего трудоёмкой ручной работы. Расширить возможности, автоматизировать и упростить процесс анализа позволяет интеграция информационной системы с BI платформой.

Было проведено сравнение инструментов платформы Pentaho BI. В качестве констрктора отчётов был выбран инструмент Saiku analytics. Данный OLAP-анализатор обладает простотой реализации REST-сервиса и высокой скоростью работы, даже при обработке огромных массивов данных.

Таким образом, необходимо разработать модуль интеграции CRM-системы Vtiger с платформой Pentaho BI Suite, а именно OLAP-анализатором Saiku, для получения возможности использования OLAP-технологий в системе Vtiger.

# 2. Постановка задачи

Необходимо разработать модуль для системы Pentaho BI Suite, для предоставления доступа CRM-платформе Vtiger к аналитическим отчётам, находящихся в хранилище Pentaho BI, и получения данных из этих отчётов.

Плагин должен:

1. Получать список сохранённых отчётов на сервере Pentaho Bi;
2. Отображать выбранный отчёт на сервере Vtiger;
3. Предоставлять данные отчёта по REST-протоколу.

Необходима простота использольвания для пользователя - возможность установки плагина без дополнительных приложений, а также интиутивно-понятный интерфейс.

Конечная версия модуля должна представлять из себя REST-сервис для получения отчётов с возможностью передачи выбранных из отчёта данных.

# 3. Архитектура системы и выбор средств разработки

Для расширения возможностей анализа CRM-системы с помощью интеграции платформы Pentaho, необходимо предоставить доступ к результатам OLAP-анализа Pentaho BI.

## 3.1. Получение отчёта с сервера Pentaho

Для начала, необходимо получить отчёт с сервера. Самый простой способ - выгрузка вручную. В пользовательской консоли пользователь может осуществлять взаимодействие с файлами. Пример такого взаимодействия демонстрирует рисунок 3.1.

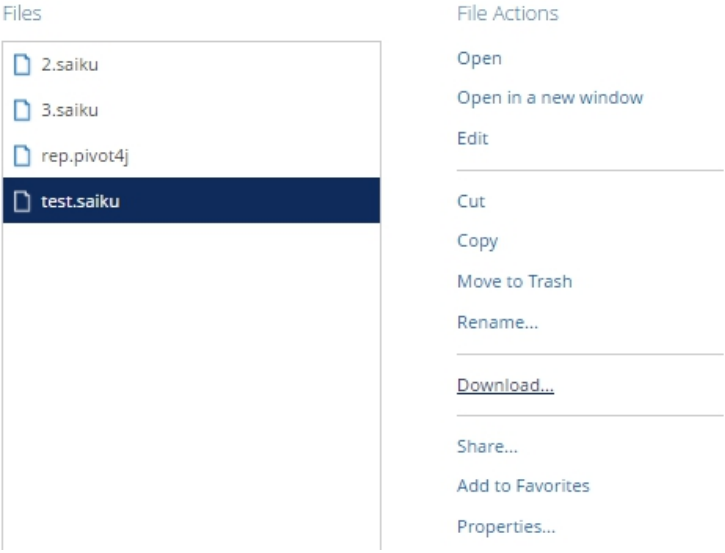


Рисунок 3.1. Загрузка отчёта с помощью консоли

Другой способ - это Kettle трасформации. Kettle или Pentaho Data Integration - это компонент платформы Pentaho, отвечающий за процесс Извлечения, Преобразования и Выгрузки данных (ETL). С по-

мощью таких операций можно осуществлять передачу данных между клиентом и сервером. Однако, Kettle требует изучения.

Ещё один способ - использование Pentaho API [8]. В нём имеется набор точек входа, URL для доступа к ресурсам сервера, позволяющих работать с источниками данных и файлами, находящимися на сервере. Пример URL для получения списка файлов и возвращаемый список можно увидеть в листинге 3.3.

Листинг 3.1. "Команда получения списка файлов выбранной директории Pentaho API"

```
1 GET /pentaho/api/repo/files/:home:admin/children
```

Сам список файлов передаётся в формате XML. Пример списка представлен в листинге 3.2.

Листинг 3.2. "Пример файлового древа"

```
1  <repositoryFileTreeDto>
2    <children>
3      <file>
4        <createdDate>1405356318621</createdDate>
5        <fileSize>-1</fileSize>
6        <folder>true</folder>
7        <hidden>false</hidden>
8        <id>fileId;/id>
9        <locale>en</locale>
10       <locked>false</locked>
11       <name>admin</name>
12       <ownerType>-1</ownerType>
13       <path>/path/to/dir</path>
14       <title>admin</title>
15       <versioned>false</versioned>
16     </file>
17   </children>
18 </repositoryFileTreeDto>
```

В параметрах запроса можно добавить фильтр. Пример запроса с фильтром приведём в листинге **??**.

Листинг 3.3. "Команда получения списка файлов выбранной директории Pentaho API"

```
1 GET /pentaho/api/repo/files/:home:admin/children?filter=*.saiku
```

Плагины в Pentaho могут выполнять следующие функции:

1. Обслуживание другого приложения;
2. Визуализация данных в новом формате;

3. Регистрация новых управляющих файлов в движке и в пользовательской консоли;

4. Интегрирование BI сервера со сторонними приложениями;

5. Изменение пользовательской консоли.

Для интеграции Vtiger CRM с Pentaho BI в данном случае лучше всего подойдёт использование Pentaho API.

## 3.2. REST API сервис

REST (сокр. от англ. Representational State Transfer — «передача состояния представления») — это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб.[9]

REST приложения используют методы протокола HTTP:

- GET - возвращает выбранный ресурс;
- POST - создаёт ресурс на сервере;
- PUT - обновляет указанный ресурс. Также используется в качестве аналога POST для создания;
- DELETE - удаляет указанный ресурс.

Pentaho API использует модель REST и технологию точек входа для доступа к данным. Точки входа позволяют соотнести URI и нужный HTTP метод. Иными словами, - это URL-ссылка, которую приложение-клиент будет использовать для общения с REST-сервисом.

На рисунке 3.2 приведена архитектура разрабатываемого приложения. Пользователь посредством web-интерфейса выбирает имеющийся на сервере Pentaho отчёт. После чего, приложение получает отчёт с сервера и передаёт его на REST-сервис. С этого момента к REST-сервису можно обращаться для получения ссылки на элементы таблицы, указанные в запросе. Пользователь, выделив поля заголовков, может сгенерировать ссылку на выделенные данные.

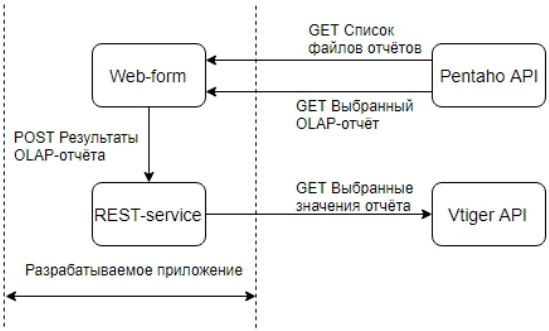


Рисунок 3.2. Архитектура разрабатываемого приложения

## 3.3. Выбор средств разработки

Для разработки сервиса предоставлено несколько воможностей: разработка на языке платформы - язык Java и разработка с помощью Pentaho Plugin Builder(SPARKL).

### 3.3.1. Java

Разработка плагина на языке Java ведётся на основе фреймворков Spring и JAX-RS. При помощи этих инструментов возможно разработать REST-приложение, основанное на аннотациях и XML-конфигурации зависимостей классов.

Необходимы навыки программирования на языке Java, понимание технологии Модель-Представление-Контроллер. При этом, нет необходимости в изучении CTools и DI. Это также даёт возможность работать в более удобной разработчику среде разработки.

Плагин, написанный на Java, обычно является REST-сервисом, запускающимся при старте сервера Pentaho и работающем на том же домене. Такому сервису задаётся URL-адрес, и в момент обращения к точке входа плагина сервер принимает запрос, идентифицирует пользователя и сам обращается к сервису.

### 3.3.2. Sparkl

Sparkl, также являясь плагином, позволяет создавать плагины и приложения для Pentaho. Разработанный на Sparkl плагин является объединением наборов CDE(приборных панелей) и Kettle работ и трансформаций.

Достоинством такого способа является прямое взаимодействие с данными и файлами сервера, что обеспечивает высокую скорость работы при обработке больших объёмов данных. Однако, для создания плагина необходимы знания Pentaho Data Integration и умение работы с контрольными панелями.

## 3.4. Итоги

Были рассмотрены решения для создания плагина Pentaho BI. Разработка на языке Java является более предпочтительным выбором разработчику ПО, так как не требует дополнительных инструментов и знаний со стороны BI. Java также удовлетворяет требованиям минимизации необходимых для установки средств платформы Pentaho.

Sparkl, в свою очередь, требует изучения таких средств и инструментов, как CDE и Kettle. Однако, данный вариант будет более подходящим вариантом людям, не имеющим опыта разработки ПО.

# 4. Проектирование и разработка системы

Разработка плагина производилась на языке Java с использованием REST API функций в качестве точек входа. Для реализации веб-интерфейса использовался HTML5 с Javascript и библиотекой Jquery.

## 4.1. Функциональность

### 4.1.1. Выбор отчёта

Для списка отчётов отправляется HTTP GET-запрос на сервер Pentaho. Для получения отчёта пользователю необходимо выбрать его из списка. После выбора отчёта отправляется GET-запрос на получение отчёта. Отчёт предоставляется в формате JSON и содержит массив данных, а также количество строк и столбцов.

В случае удачного подключения, полученные данные отправляются REST-сервису для дальнейшей возможности работать с ними извне, и генерируется таблица. В случае неудачного подключения, пользователю выводится соответствующее сообщение об ошибке. Такими ошибками могут быть:

1. Недоступность сервера;

2. Неудачная попытка загрузки отчёта.

### 4.1.2. Работа с полученными данными

При получении GET-запроса на свой URL-адрес, REST-сервис обрабатывает его параметры и по ним собирает массив данных, который отправляется в ответе на запрос.

Веб-интерфейс даёт возможность сгенерировать URL для доступа к желаемым элементам по REST-протоколу. Для начала пользователю необходимо выбрать данные при помощи выделения необходимых заголовков таблицы кликом мыши. Для генерации адреса необходимо нажать на кнопку "Generate URL". Также есть возможность вернуться в меню выбора отчёта кнопкой "Back".

## 4.2. Реализация

Главная часть приложения состоит из работы с параметрами HTTP-запроса и создания массива данных по полученным параметрам.

### 4.2.1. Структура компонентов и описание

Приложение состоит из 2 классов. Ниже приведена таблица с этими классами и их описанием. В таблице 4.1

| Название класса | Описание |
|---|---|
| PentahoDataIntegratorService | Описание методов REST сервиса |
| DataSelector | Класс-анализатор, выдающий выборку из массива по параметрам запроса |

Таблица 4.1. Классы и их описание

### 4.2.2. REST-сервис

В языке Java определена поддержка REST через Java Specification Request (JSR) 311.[10] Эта спецификация называется JAX-RS. Для реализации REST-сервиса в данной работе использовалась библиотека Jersey. Jersey предоставляет классы для имплементации веб-сервера в виде сервлета. Конфигурация сервиса находится в отдельном файле - plugin.spring.xml.

Базовый адрес сервиса:

Листинг 4.1. "Базовый адрес сервиса"

```
1 http://your_domain:port/display-name/url-pattern/path_from_rest_class
```

Сервер анализирует входящий HTTP запрос и выбирает соответствующий ему класс и метод для ответа. Выбор осуществляется на основе аннотаций в классах и методах. Используемые в работе аннотации представлены в таблице 4.2:

| Аннотация | Описание |
|---|---|
| @PATH(your_path) | Устанавливает путь на конкатенацию адреса домена и адреса метода |
| @GET | Метод является HTTP GET запросом |
| @Produces(MediaType) | Какой тип данных будет являться ответом на запрос |
| @QueryParam | Какие параметры будут внесены в запрос |

Таблица 4.2. Основные аннотации, используемые в работе

В классе, представленном в листинге 4.2, метод будет доступен по адресу base/pattern/export. Он будет являться GET запросом и отправлять ответ пользователю.

Листинг 4.2. "GET запрос сервиса"

```
1 @Path("@plugin.java.rest.path.root@")
2 @GET
3 @Path("/export")
4 @Produces(MediaType.APPLICATION_JSON)
5 public Response getSomeData {
6   @QueryParam("heads") String headers;
7 ...
8 }
```

Данный класс-ресурс будет доступен по адресу localhost:8080/pentaho/plugin/api/data-integrator/export?heads=chosen_headers_ при запросе типа GET, и пользователю будет возвращаться выборка массива по данным заголовкам в виде JSON строки.

### 4.2.3. Графический интерфейс

При проектировании графического интерфейса уклон был сделан на интуитивность и скорость работы.

Для реализации использовалась технология HTML и язык Javascript, а также библиотека Jquery. На рисунках 4.1 и 4.2 показано главное окно, которое открывается пользователю при выборе владки плагина и окно сгенерированного отчёта. За дизайн отвечает css код, находящийся в отдельном файле.

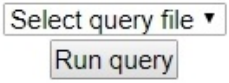


Рисунок 4.1. Стартовое окно интерфейса плагина

Стартовая страница выглядит как селектор с кнопкой подтверждения.

Generate URL | Here will be URL | back

| | USA | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CA | | | | OR | | WA | | | | | | |
| | Beverly Hills | Los Angeles | San Diego | San Francisco | Portland | Salem | Bellingham | Bremerton | Seattle | Spokane | Tacoma | Walla Walla | Yakima |
| Product Family | Store 6 | Store 7 | Store 24 | Store 14 | Store 11 | Store 13 | Store 2 | Store 3 | Store 15 | Store 16 | Store 17 | Store 22 | Store 23 |
| Drink | 590,419 | 850,181 | 746,353 | 105,698 | 324,723 | 1 732,508 | 141,182 | 777,301 | 990,566 | 665,289 | 1 822,461 | 34,357 | 437,315 |
| Food | 3 635,132 | 8 725,71 | 8 905,011 | 662,178 | 2 696,758 | 11 030,067 | 696,042 | 8 760,093 | 8 596,461 | 3 901,889 | 11 113,203 | 983,866 | 3 661,138 |
| Non-Consumable | 907,346 | 2 059,027 | 2 199,299 | 220,944 | 1 021,478 | 3 388,466 | 212,235 | 2 122,231 | 1 930,098 | 1 214,785 | 3 165,165 | 75,546 | 1 175,884 |

Рисунок 4.2. Пример сгенерированной таблицы

На окне работы с отчётом выше таблицы имеются 3 элемента:

- Кнопка "Назад"для возвращения обратно на стартовую страницу;
- Кнопка "generateURL"для генерации URL по выделенным заголовкам таблицы;
- Поле, в которое выводится сгенерированный URL.

Реализацию Javascript функций можно найти в приложении 2. В таблице 4.3 указаны основные функции.

| Функция | Комментарий |
|---|---|
| getFiles() | Функция получения списка файлов аналитических отчётов Pentaho BI |

| | |
|---|---|
| getData() | Функция загрузки данных выбранного отчёта |
| createTable() | Функция генерации таблицы по полученным данным |
| generateURL() | Функция создания URL на основе выбранных данных |

Таблица 4.3. Основные функции Javascript-части программы

## 4.3. Итоги

Было разработано приложение-плагин для Pentaho BI server, в частности, для Pentaho User Console. Данное приложение реализует интеграцию модуля OLAP-анализа платформы Pentaho в CRM-систему посредством предоставления доступа к уже готовым отчётам, хранящимся на сервере Pentaho BI.

# 5. Тестирование системы

Тестирование было разделено на **backend**(внутренняя реализация) и **frontend**(внешнее представление) части.

## 5.1. Backend тестирование

В тестировании класса выборки данных из массива реализован следующий алгоритм:

1. Сгенерирована JSON строка;
2. Сгенерирован JSON массив в соответствии со строкой;
3. Произведено сравнение полученного результата с этелонной моделью, указанной вручную.

Для тестирования класса выборки данных использовался фреймворк JUnit.

Был создан класс TestDataSelector, часть которого представлена в листинге 5.1:

Листинг 5.1. "Класс тестирования"

```
1  import org.junit.Test;
2  import static org.junit.Assert.assertEquals;
3
4  public class TestDataSelector {
5
6    @Test
7    public void selectDataTest() {
8      String json = "{[[{\"value\":\"null\",\"type\":\"COLUMN_HEADER
           \"}, {\"value\":\"USA\",\"type\":\"COLUMN_HEADER\"},
9      ...
10     {\"value\":\"6532\",\"type\":\"DATA_CELL\"}, {\"value
           \":\"4387\",\"type\":\"DATA_CELL\"}]]}";
11
12     JSONObject jsonObj = new JSONObj();
13     JSONArray array = new JSONArray();
14     DataSelector dataSelector = new DataSelector(json);
15
16     jsonObj.put(value, "Store 6");
17     jsonObj.put(type, "COLUMN_HEADER");
18     array.add(jsonObj);
19     ...
20     assertEquals(dataSelector.selectData("Store 6|Drinks"), array);
21   }
22   ...
23 }
```

## 5.2. Тестирование REST-сервиса

Jersey, используемый в проекте для тестирования предоставляет минимальный веб-сервер для развёртывания пользователем. Для тестирования разработанного сервиса необходимо проверить работу точек входа для получения массива данных и для отправки отчёта в формате JSON.

Для закрытия сервера после исполнения всех запланированных тестов использовалась аннотация @AfterClass.

Листинг 5.2. "Использование Аннотации AfterClass"

```
1 @AfterClass
2 public static void afterClass() throws Exception {
3   server.close();
4 }
```

Для верификации результата теста нужно сравнить код ответа от сервера с ожидаемым(200 - OK), указанным в спецификации.

Листинг 5.3. "Сравнение ответного кода с ожидаемым положительным"

```
1 assertEquals(Response.Status.OK.getStatusCode(), response.getStatus
      ());
```

Были протестированы варианты кодов ответа: позитивного(200), негативного(505) и случай недоступности сервиса(404).

## 5.3. Frontend тестирование

Тестирование интерфейса плагина, в данном случае, представляет из себя тестирование Javascript кода, отвечающего за выбор опции выпадающего списка и обработчиков нажатия на кнопку. Для тестирования использовалась библиотека Jasmine, предоставляющая возможность создавать “заглушки” для элементов:

Листинг 5.4. "Пример использования библиотеки Jasmine"

```
1 var exec;
2 beforeEach(){
3   exec = jasmine.createSpy();
4   spyOn(document 'getElemntById').andReturn({exec:getTable})
5 }
```

## 5.4. Итоги

Тестирование любого проекта является его важной частью. Разделение тестирования на backend и frontend части позволяет проверить работу как в серверной части, так и в части интерфейса.

Тесты для серверной части помогают проверить корректность выборки данных из JSON массива.

Данные, полученные в тестировании, свидетельствуют о корректной работе сервиса. При некоректно выбранных данных генерация не происходит и пользователю возвращается корректная ошибка.

# 6. Анализ разработанной системы

В результате разработки была реализована возможность интеграции аналитических отчётов платформы бизнес-аналитики в CRM-систему Vtiger.

Интеграция происходит с помощью веб-сервиса, находящегося на сервере Pentaho. С помощью приложения возможно передать данные из отчёта OLAP-аналитики с сервера Pentaho CRM-системе.

По итогам исследования было выяснено, что инструменты аналитики CRM Vtiger недостаточно мощны для комплексного анализа. Реализованное приложение даёт возможность предоставить результаты работы OLAP-анализатора Saiku платформы Pentaho BI по REST-протоколу.

Продемонстрируем работу созданного приложения. У нас уже имеется аналитический отчёт, созданный в Saiku и открытый рассматриваемым плагином. Отсёт продемонстрирован на рисунке 6.1 С этого момента уже возможно получить доступ к данным в отчёте, введя URL-адрес REST-сервиса с корректными параметрами.

Generate URL | Here will be URL | back

| | USA | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CA | | | | OR | | WA | | | | | | |
| | Beverly Hills | Los Angeles | San Diego | San Francisco | Portland | Salem | Bellingham | Bremerton | Seattle | Spokane | Tacoma | Walla Walla | Yakima |
| Product Family | Store 6 | Store 7 | Store 24 | Store 14 | Store 11 | Store 13 | Store 2 | Store 3 | Store 15 | Store 16 | Store 17 | Store 22 | Store 23 |
| Drink | 590,419 | 850,181 | 746,353 | 105,698 | 324,723 | 1 732,508 | 141,182 | 777,301 | 990,566 | 665,289 | 1 822,461 | 34,357 | 437,315 |
| Food | 3 635,132 | 8 725,71 | 8 905,011 | 662,178 | 2 696,758 | 11 030,067 | 696,042 | 8 760,093 | 8 596,461 | 3 901,889 | 11 113,203 | 983,866 | 3 661,138 |
| Non-Consumable | 907,346 | 2 059,027 | 2 199,299 | 220,944 | 1 021,478 | 3 388,466 | 212,235 | 2 122,231 | 1 930,098 | 1 214,785 | 3 165,165 | 75,546 | 1 175,884 |

Рисунок 6.1. Отчёт в разработанном плагине

Выберем данные для генерации URL-адреса и сгенерируем его, нажав на соответствующую кнопку в интерфейсе. Результат демонстрирует рисунок 6.2.

Generate URL | a-integrator/api/expo | back

| | USA | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CA | | | | OR | | WA | | | | | | |
| | Beverly Hills | Los Angeles | San Diego | San Francisco | Portland | Salem | Bellingham | Bremerton | Seattle | Spokane | Tacoma | Walla Walla | Yakima |
| Product Family | Store 6 | Store 7 | Store 24 | Store 14 | Store 11 | Store 13 | Store 2 | Store 3 | Store 15 | Store 16 | Store 17 | Store 22 | Store 23 |
| Drink | 590,419 | 850,181 | 746,353 | 105,698 | 324,723 | 1 732,508 | 141,182 | 777,301 | 990,566 | 665,289 | 1 822,461 | 34,357 | 437,315 |
| Food | 3 635,132 | 8 725,71 | 8 905,011 | 662,178 | 2 696,758 | 11 030,067 | 696,042 | 8 760,093 | 8 596,461 | 3 901,889 | 11 113,203 | 983,866 | 3 661,138 |
| Non-Consumable | 907,346 | 2 059,027 | 2 199,299 | 220,944 | 1 021,478 | 3 388,466 | 212,235 | 2 122,231 | 1 930,098 | 1 214,785 | 3 165,165 | 75,546 | 1 175,884 |

Рисунок 6.2. Выбор данных для генерации URL

Далее можно использовать данный URL для дальнейшего использования результатов работы аналитического отчёта. Для этого создадим страницу с параграфом, что поместить туда полученный json-объект. Пример показан на рисунке 6.3

| | CA | CA | CA | CA |
|---|---|---|---|---|
| | Beverly Hills | Los Angelos | San Diego | San Francisco |
| | Store 6 | Store 7 | Store 24 | Store 14 |
| Drinks | 590,419 | 850,181 | 746,353 | 105,698 |

Рисунок 6.3. Демонстрация REST-сервиса

Таким образом, можно сказать, что задача выполнена. С помощью разработанного плагина можно посредством REST-протокола предоставлять доступ к ресурсам отчётов системы Pentaho BI Suite.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения работы, были исследованы информационные системы, хранящие и позволяющие анализировать информацию, связанную с бизнес-процессами и менеджментом.

Исследование показало, что наибольшую эффективность представляет связка информационная система - платформа бизнес-аналитики. Такая связка даёт возможность использовать инструменты BI платформы для обработки данных из хранилища CRM.

В данной работе исследовалась возможность интеграции CRM-системы Vtiger и BI платформы Pentaho BI Suite.

Для интеграции был написан плагин для Pentaho, дающий возможность получить необходимые данные из OLAP-отчёта. Также, разработанный плагин выводит отчёт и генерирует URL-адрес по выбранным заголовкам таблицы.

В процессе разработки возникли определённые трудности. Так, на момент разработки плагина, не удавалось получить URL для доступа к отчёту по причине нерабочей документации Saiku(ошибка 404). На поиск решения, либо замены на другой вариант, ушло слишком много времени, что впоследствии стало причиной превышения сроков выполнения работы.

Разработанный плагин действительно расширяет возможности бизнес-аналитики по сравнению со встроенными в CRM-систему инструментами. При использовании модуля появляются возможности получать результаты инструмента аналитики и прогнозирования Saiku.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. What is CRM [Электронный ресурс], TechTarget. — URL: http://searchcrm.techtarget.com/definition/CRM (дата обращения: 27.06.2017).
2. Salesplatform Wiki [Электронный ресурс], SalesPlatform. — URL: https://goo.gl/EFSpny (дата обращения: 27.06.2017).
3. Vtiger Documentation [Электронный ресурс], Vtiger. — URL: http://community.vtiger.com/help/ (дата обращения: 27.06.2017).
4. BI-системы [Электронный ресурс], Норбит. — URL: http://www.norbit.ru/products/groups/189.html (дата обращения: 28.06.2017).
5. Pentaho Reporting. [Электронный ресурс], Rittan Holdings Ltd. — URL: http://community.pentaho.com/projects/reporting/ (дата обращения: 28.06.2017).
6. Pivot4j [Электронный ресурс], Pivot4j Team. — URL: http://www.pivot4j.org/ (дата обращения: 28.06.2017).
7. Saiku analytics vs Pivot4j [Электронный ресурс], Stack Overflow. — URL: https://stackoverflow.com/questions/27000958/saiku-analytics-vs-pivot4j (дата обращения: 28.06.2017).
8. API Documentation for BA Platform [Электронный ресурс], Pentaho Corporation. — URL: https://help.pentaho.com/\T2A\cyro\T2A\cyrb\T2A\cyrr\T2A\cyra\T2A\cyrshch\T2A\cyre\T2A\cyrn\T2A\cyri\T2A\cyrya: 28.06.2017).
9. Архитектура REST [Электронный ресурс], ХабраХабр. — URL: https://habrahabr.ru/post/38730/ (дата обращения: 29.06.2017).
10. JSR Overview [Электронный ресурс], Java Community Process. — URL: https://jcp.org/en/jsr/overview (дата обращения: 30.06.2017).

# ПРИЛОЖЕНИЕ А
# **Листинги.**

Листинг А.1. main.html

```
<html>
<head>
    <meta charset='utf-8'>
    <script type='text/javascript' src='main.js'></script>
    <script type='text/javascript' src='jquery-3.2.1.min.js'></
        script>
    <link rel='stylesheet' type='text/css' href='main.css'/>
</head>
<body>
    <div class='center'>
        <select id='files'>
            <option value='-1'>Select query file</option>
        </select>
        <button class='center' id='gen' onclick='generateURL()'
            style='display: none'>Generate URL</button>
        <input type='text' id='url' class='center' size='15' value='
            Here will be URL' style='display: none' />
        <button class='center' id='back' onclick='back()' style='
            display: none'>back</button>
    </div>
    <div class='center'>
        <button class='center' id='exec' onclick='getTable()'>Run
            query</button>
        <table id='data'></table>
    </div>
   <script type='text/javascript'>
        getFiles();
        setCellClick();
    </script>
</body>
</html>
```

Листинг А.2. main.js

```
const restServiceGET = '/pentaho/plugin/data-integrator/api/export?'
    , // Адрес для GET-запроса
    REST-сервиса
    restServicePUT = '/pentaho/plugin/data-integrator/api/import',
        // Адрес для PUT-запроса
        REST-сервиса
    saiku = '/pentaho/plugin/saiku/api/admin/export/saiku/json?file
        =/home/admin/', // Адрес файла отчёта (без имени
        файла)
    filepath = '/pentaho/api/repo/files/:home:admin/children?filter
        =*.saiku'; // Адрес папки с
        отчётами
var json; // JSON-файл, хранящий данные отчёта.

// Функция получения списка файлов.
```

```
8  function getFiles() {
9      var request = new XMLHttpRequest(),
10         func = function f(e) {
11         if (request.readyState === 4 && request.status === 200) {
12             var parser = new DOMParser(),
13                 xmlDoc = parser.parseFromString(request.responseText
                       , 'text/xml'),
14                 names = xmlDoc.getElementsByTagName('name');
15
16             for (i of names) {
17                 $('#files').append('<option>'+i.childNodes[0].
                       nodeValue+'</option>');
18             }
19         }
20     };
21
22     request.open('GET', filepath);
23     request.onload = func;
24     request.send();
25 }
26
27 // Метод отправки отчёта REST-сервису
28 function sendJSON() {
29   var request = new XMLHttpRequest();
30   request.open('PUT', restServicePUT);
31   request.setRequestHeader('Content-Type', 'application/json;
         charset=UTF-8');
32   request.send(JSON.stringify(json));
33 }
34
35 // Функция получения отчёта.
36 function getData(filename) {
37     var rootpath = saiku + filename,
38     request = new XMLHttpRequest();
39
40     request.open('GET', rootpath);
41     request.onload = function (e) {
42         if (request.readyState === 4 && request.status === 200) {
43             json = JSON.parse(request.responseText);
44             sendJSON();
45             createTable();
46         }
47     };
48     request.send();
49 }
50
51 // Метод генерации URL
52 function generateURL() {
53     var url = window.location.host + restService,
54         rows = document.getElementById('data').rows;
55
56     for (i = 0; i < getColHeadNum(); i++) {
57         for (j = getRowHeadNum(); j < rows[i].cells.length; j++) {
58             if (rows[i].cells[j].classList.contains('click')) {
59                 url += rows[i].cells[j].innerHTML + '|';
60             }
61         }
62     }
```

```
63
64      for (i = getColHeadNum()-1; i < rows.length; i++) {
65          for (j = 0; j < getRowHeadNum(); j++) {
66              if (rows[i].cells[j].classList.contains('click')) {
67                  url += rows[i].cells[j].innerHTML + '|';
68              }
69          }
70      }
71
72      url = url.slice(0, -1);
73
74      document.getElementById('url').value = url;
75  }
76
77  // Функция генерации таблицы.
78  function createTable() {
79      var table = document.getElementById('data'),
80          tr,
81          td,
82          val = '',
83          colspan = 1,
84          same = false,
85          cellset = json.cellset;
86
87      for (i = 0; i < json.height; i++) {
88          tr = table.insertRow();
89          for (j = 0; j < json.width; j++) {
90              if (cellset[i][j].type === 'DATA_CELL') {
91                  td = tr.insertCell();
92                  td.appendChild(document.createTextNode(cellset[i][j
                        ].value));
93              }else if (cellset[i][j].value === 'null') {
94                  td = tr.insertCell();
95                  td.className = 'empty';
96              } else if (cellset[i][j].value !== val) {
97                  if (colspan > 1) {
98                      th.colSpan = colspan;
99                      th.id = j-colspan+1;
100                     colspan = 1;
101                 }
102                 val = cellset[i][j].value
103                 th = document.createElement('th');
104                 tr.appendChild(th);
105                 th.appendChild(document.createTextNode(val));
106             } else {
107                 colspan++;
108                 if (j === json.width - 1) {
109                     th.colSpan = colspan;
110                     colspan = 1;
111                 }
112             }
113         }
114     }
115
116     for (j = getRowHeadNum()-2; j >= 0; j--) {
117         for (i = getColHeadNum(); i < json.height-1; i++) {
118             if (cellset[i][j].value === cellset[i+1][j].value) {
119                 if (rowspan === 1) {
```

```
120                     rowspan++;
121                     continue;
122                 }
123                 rowspan++;
124                 if (i === json.height-2) {
125                     table.rows[i+1].deleteCell(j);
126                     table.rows[i-rowspan+2].cells[j].rowSpan =
                            rowspan;
127                     rowspan = 1;
128                 }
129                 table.rows[i].deleteCell(j);
130             } else {
131                 if (rowspan !== 1) {
132                     table.rows[i-rowspan+1].cells[j].rowSpan =
                            rowspan;
133                     table.rows[i].deleteCell(j);
134                     rowspan = 1;
135                 }
136             }
137         }
138     }
139 }
140
141 // Функция, которая делает заголовки кликалъбельными.
142 // При клике на заголовок добавляется/убирается css класс click.
143 function setCellClick() {
144     var func = function f() {
145         this.classList.toggle('click');
146     };
147     $('#data').on('click', 'th', func);
148 }
```

Листинг А.3. main.css

```
1  /* Класс для отображения элементов по-центру */
2  .center {
3      text-align: center;
4      font-size: 14pt;
5      padding: 0.1%;
6  }
7
8  /* Класс ячеек, на которые нажали */
9  .click {
10     background: #D0E0FF;
11 }
12
13 /* Класс для пустых ячеек в заголовках таблицы */
14 .empty {
15     background: #FFFFFF;
16     border: 0;
17 }
18
19 select {
20     font-size: 14pt;
21     text-align-last: center;
22 }
23
```

```
table {
    margin: auto;
    border: 4px #A9A9A9;
    border-collapse: collapse;
}

th, td {
    text-align: center;
    padding: 5px;
    border: 1px solid #A9A9A9;
}

th {
    font-weight: normal;
    background: #F5F5F5;
}
```