

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики (ИК)
Направление подготовки 09.04.03 Прикладная информатика
Кафедра программной инженерии (ПИ)

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Исследование и разработка методов формирования динамических отчетных документов в корпоративных web-порталах

УДК 004.774.6:657.478.24

Студент

Группа	ФИО	Подпись	Дата
8KM51	Сенина Анастасия Анатольевна		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор кафедры «Программная инженерия» ТПУ	Тузовский Анатолий Федорович	д.т.н., профессор		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. МЕН.	Баннова К. А.	Кандидат экономических наук		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ЭБЖ	Пустовойтова М. И.	Кандидат химических наук		

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
Программная инженерия	Иванов М. А.	Кандидат технических наук		

Запланированные результаты обучения по ООП

Код результата	Результат обучения (выпускник должен быть готов)
<i>ПРОФЕССИОНАЛЬНЫЕ компетенции</i>	
ПК-1	Способен использовать нормативные правовые документы в профессиональной деятельности.
ПК-2	Способен при решении профессиональных задач анализировать социально-экономические проблемы и процессы с применением методов системного анализа и математического моделирования.
ПК-3	Способен использовать основные законы естественнонаучных дисциплин в профессиональной деятельности и эксплуатировать современное электронное оборудование и информационно-коммуникационные технологии в соответствии с целями образовательной программы бакалавра.
ПК-4	Способен ставить и решать прикладные задачи с использованием современных информационно-коммуникационных технологий.
ПК-5	Способен осуществлять и обосновывать выбор проектных решений по видам обеспечения информационных систем.
ПК-6	Способен документировать процессы создания информационных систем на всех стадиях жизненного цикла.
ПК-7	Способен использовать технологические и функциональные стандарты, современные модели и методы оценки качества и надежности при проектировании, конструировании и отладке программных средств.
ПК-8	Способен проводить обследование организаций, выявлять информационные потребности пользователей, формировать требования к информационной системе, участвовать в реинжиниринге прикладных и информационных процессов.
ПК-9	Способен моделировать и проектировать структуры данных и знаний, прикладные и информационные процессы.
ПК-10	Способен применять к решению прикладных задач базовые алгоритмы обработки информации, выполнять оценку сложности алгоритмов, программировать и тестировать программы.
ПК-11	Способен принимать участие в создании и управлении ИС на всех этапах жизненного цикла.
ПК-12	Способен эксплуатировать и сопровождать информационные системы и сервисы.
ПК-13	Способен принимать участие во внедрении, адаптации и настройке прикладных ИС.
ПК-14	Способен принимать участие в реализации профессиональных коммуникаций в рамках проектных групп, презентовать результаты проектов и обучать пользователей ИС.
ПК-15	Способен проводить оценку экономических затрат на проекты по информатизации и автоматизации решения прикладных задач.

ПК-16	Способен оценивать и выбирать современные операционные среды и информационно-коммуникационные технологии для информатизации и автоматизации решения прикладных задач и создания ИС.
ПК-17	Способен применять методы анализа прикладной области на концептуальном, логическом, математическом и алгоритмическом уровнях.
ПК-18	Способен анализировать и выбирать методы и средства обеспечения информационной безопасности.
ПК-19	Способен анализировать рынок программно-технических средств, информационных продуктов и услуг для решения прикладных задач и создания информационных систем.
ПК-20	Способен выбирать необходимые для организации информационные ресурсы и источники знаний в электронной среде.
ПК-21	Способен применять системный подход и математические методы в формализации решения прикладных задач.
ПК-22	Способен готовить обзоры научной литературы и электронных информационно-образовательных ресурсов для профессиональной деятельности.
<i>Универсальные компетенции</i>	
ОК-1	Способен использовать, обобщать и анализировать информацию, ставить цели и находить пути их достижения в условиях формирования и развития информационного общества.
ОК-2	Способен логически верно, аргументированно и ясно строить устную и письменную речь, владеть навыками ведения дискуссии и полемики.
ОК-3	Способен работать в коллективе, нести ответственность за поддержание партнерских, доверительных отношений.
ОК-4	Способен находить организационно-управленческие решения и готов нести за них ответственность.
ОК-5	Способен самостоятельно приобретать и использовать в практической деятельности новые знания и умения, стремится к саморазвитию.
ОК-6	Способен осознавать социальную значимость своей будущей профессии, обладать высокой мотивацией к выполнению профессиональной деятельности.
ОК-7	Способен понимать сущность и проблемы развития современного информационного общества.
ОК-8	Способен работать с информацией в глобальных компьютерных сетях.
ОК-9	Способен свободно пользоваться русским языком и одним из иностранных языков на уровне, необходимом для выполнения профессиональных задач.
ОК-10	Способен использовать методы и средства для укрепления здоровья

	и обеспечения полноценной социальной и профессиональной деятельности.
ОК-11	Способен уважительно и бережно относиться к историческому наследию и культурным традициям, толерантно воспринимать социальные и культурные различия.
ОК-12	Способен использовать Гражданский кодекс Российской Федерации, правовые и моральные нормы в социальном взаимодействии и реализации гражданской ответственности.
ОК-13	Способен понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности, в том числе защиты государственной тайны.
ОК-14	Способен применять основные методы защиты производственного персонала и населения от возможных последствий аварий, катастроф, стихийных бедствий, технику безопасности на производстве.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики (ИК)
Направление подготовки 09.04.03. Прикладная информатика
Кафедра программной инженерии (ПИ)

УТВЕРЖДАЮ:
Зав. кафедрой

(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

В форме:
Магистерской диссертации

Студенту:

Группа	ФИО
8KM51	Сенина Анастасия Анатольевна

Тема работы:

Исследование и разработка методов формирования динамических отчетных документов в корпоративных web-порталах	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	
<i>(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).</i>	Объект исследования – методы динамического формирования отчетов в корпоративных web-порталах. Объект разработки – корпоративный веб-портал «SmartReports» с динамическим формированием алгоритмов по обработке данных и шаблонов отчетных документов. Для разработки программного средства была использована среда разработки Visual Studio 2015. Веб-приложение создано с помощью технологии разработки ASP.NET MVC4 на языке C#. Характеристика режима работы – непрерывный. Для эксплуатации веб-

	<p>приложения потребуется веб-браузер, поддерживающий HTML5 и CSS3. Эксплуатация веб-приложения с точки зрения безопасности окружающей среды является безвредной. Корпоративный веб-портал «SmartReports» является эффективным с точки зрения экономической эффективности, ресурсоэффективности и ресурсосбережения. Работа пользователя осуществляется через интернет.</p>
<p>Перечень подлежащих исследованию, проектированию и разработке вопросов</p> <p><i>(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).</i></p>	<p>Задача исследования состоит в выявлении методов построения отчетной документации, а также в разработке корпоративного веб-портала с применением динамического метода построения отчетных документов и обработки данных.</p> <p>Процедура исследования включает в себя обзор существующих систем по формированию отчетных документов, обзор технологий разработки веб-приложений и выбор наиболее подходящей, проектирование архитектура системы (раскрытие шаблона разработки MVC, архитектуры трехслойной системы), проектирование системы (описание подсистем, построение статических и динамических UML диаграмм, концептуальной модели базы данных), разработка веб-приложения (описание реализации связи и работы с базой данных, реализации спроектированных подсистем), отладка и документирование (написание пояснительной записки).</p>
<p>Перечень графического материала</p> <p><i>(с точным указанием обязательных чертежей)</i></p>	<ol style="list-style-type: none"> 1. Алгоритмы технологий разработки веб-приложений 2. Архитектура веб-портала на основе запросов пользователя 3. Архитектура корпоративного веб-портала «SmartReports» 4. Архитектура трехслойной разработки веб-приложений 5. Статические и динамические UML диаграммы 6. Концептуальная модель базы данных 7. Пользовательский интерфейс корпоративного веб-портала «SmartReports»

Консультанты по разделам выпускной квалификационной работы <i>(с указанием разделов)</i>	
Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Баннова Кристина Алексеевна
Социальная ответственность	Пустовойтова Марина Игоревна
Названия разделов, которые должны быть написаны на русском и иностранном языках:	
Обзор технологий разработки веб-приложений	

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор кафедры «Программная инженерия» ТПУ	Тузовский Анатолий Федорович	доктор технических наук, профессор		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8KM51	Сенина Анастасия Анатольевна		

Реферат

Ключевые слова: КОРПОРАТИВНЫЙ ВЕБ-ПОРТАЛ, ОБРАБОТКА ДАННЫХ, ОТЧЕТ, ШАБЛОН, ASP.NET, MVC, ENTITY FRAMEWORK, ВЕБ-ПРИЛОЖЕНИЕ.

Объектом исследования данной магистерской диссертации являются методы анализа данных производства и построения отчетной документации с использованием корпоративных веб-порталов. Объектом разработки является корпоративный веб-портал «SmartReports» по обработке данных производства и формированию отчетной документации.

Целью разработки корпоративного веб-портала «SmartReports» является предоставление возможности анализа показателей процессов производства, повышение осведомленности сотрудников предприятия о работе предприятия.

В ходе разработки применялась технология разработки ASP.NET веб-приложений на основе MVC 4 архитектуры в среде разработки программного обеспечения Microsoft Visual Studio 2015.

Актуальность работы заключается в часто возникающей потребности решения задачи подготовки различных отчетов о деятельности предприятия для оперативного принятия решений с использованием веб-приложений.

В разделе «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение» доказана и обоснована экономическая эффективность работы, выбрано наиболее эффективное исполнение проекта.

Магистерская диссертация включает в себя: 114 страниц, 32 рисунка, 23 таблицы, 23 литературных источника, 5 приложений.

Оглавление

Введение	11
Глава 1. Обзор литературных источников	13
1.1 Обзор технологий разработки веб-приложений	13
1.2 Обзор конкурентных систем	28
Выводы по главе	29
Глава 2. Постановка и решение задачи	31
1.1 Описание задачи динамического формирования отчетов	31
1.2 Анализ методов формирования отчетной документации	32
Выводы по главе	42
Глава 3. Программная реализация методов динамического формирования отчетов	44
3.1 Цели создания веб-портала SmartReport	44
3.2 Описание веб-портала SmartReport	44
3.3 Программная реализация динамического формирования отчетов	45
3.4 Программная реализация подсистемы администрирования веб-портала	59
3.5 Модель базы данных веб-портала «SmartReports»	64
3.6 Организация работ по разработке портала SmartReports	66
Выводы по главе	66
Глава 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	67
4.1 Оценка коммерческого потенциала и перспективности проведения научных исследований и позиции ресурсоэффективности и ресурсосбережения.	67
4.2 Планирование научно-исследовательских работ.	73

4.3 Определение ресурсной, финансовой, бюджетной, социальной и экономической эффективности исследования.	82
Вывод	84
Глава 5. Социальная ответственность	86
5.1 Техногенная безопасность	86
5.2 Электробезопасность	91
5.3 Региональная безопасность	92
5.4 Организационные мероприятия обеспечения безопасности	93
5.5 Особенности законодательного регулирования проектных решений	95
5.6 Пожарная безопасность	96
Список публикаций	98
Список использованных источников	99
Заключение	101
Приложение А	102
Приложение Б	112
Приложение В	113
Приложение Г	114
Приложение Д	115

Введение

Практически каждое промышленное предприятие ведет строгий учет ряда производственных параметров, например, производительность, статьи расходов, сбои оборудования, перебои в электроснабжении и т.д. Такого рода информация должна оперативно отображаться в отчетах, так как сырые данные, т.е. значения, хранящиеся в базе данных, порой избыточны и без определенной обработки не пригодны для анализа. Так, возникает необходимость формировать отчетную документацию для оптимизации производства.

Отчетная документация позволяет выполнять конструктивный анализ бизнес-процессов с целью их оптимизации. Отчеты востребованы как на верхних уровнях руководства, так и рабочим персоналом. Таким образом, в зависимости от потребителя и вида отчета требуется производить определенную выборку из базы данных, а также обрабатывать данные тем или иным образом.

В большинстве случаев пользователь не имеет возможности самостоятельно конфигурировать параметры отчета. Для работы с существующими программными системами он составляет список требований, в соответствии с которыми программист разрабатывает программный код, формирующий отчет.

Безусловно, нет единственной общепринятой схемы обработки данных для конкретного вида отчета и для конкретного его потребителя. Возникает необходимость конфигурирования множества параметров отчетов: источники данных, алгоритм обработки данных, вид их отображения в отчете.

Актуальность работы заключается в часто возникающей потребности решения задачи подготовки различных отчетов о деятельности предприятия для оперативного принятия решений с использованием веб-приложений.

Целью магистерской диссертации является разработка методов анализа данных и алгоритмов динамического формирования шаблонов отчетных документов.

Научной новизной предлагаемого подхода является набор методов, позволяющих пользователям самостоятельно конфигурировать алгоритм обработки данных и вид их отображения в отчете. Использование разработанных алго-

ритмов позволит значительно ускорить и упростить процесс анализа производственных показателей и поспособствует процессу принятия решений. Система выступает в качестве высоко функционального инструментария по обработке и представлению данных.

Практическая значимость работы заключается в том, что разработанные методы обработки данных и алгоритмы их визуализации в виде отчетов могут быть применены в любом программном комплексе, осуществляющем хранение данных о технологических процессах и формирующем на их основе отчетную документацию. Разрабатываемое программное обеспечение не зависит от специализации предприятия, что делает его универсальным в своей области применения. Благодаря гибкой реализации системы, возможно учесть особенности предприятия (например, уникальная база данных). Это достигается путем дополнительной настройки по требованию заказчика.

Предложенное в диссертационной работе решение реализуется в составе корпоративного веб-портала «SmartReport», разрабатываемого в компании «ЭлеСи».

Данная магистерская диссертация содержит пять глав.

В первой главе проведен теоретический обзор технологий разработки веб-приложений, а также рассмотрены аналогичные системы по предоставлению отчетной документации. Во второй главе выполнен анализ двух проектируемых систем по формированию отчетных документов, обоснован выбор системы для разработки. Третья глава содержит программную реализацию корпоративного веб-приложения «SmartReports».

В четвертой главе производится обоснование целесообразности проведения исследования и разработки системы. Пятая глава содержит анализ угроз при работе с системой. Приложения содержат фрагменты программного кода, а также скриншоты системы.

Глава 1. Обзор литературных источников

1.1 Обзор технологий разработки веб-приложений

В настоящее время наиболее используемыми технологиями разработки веб-приложений являются Node.js, Ruby on Rails и ASP.Net.

1.1.1 Технология Node.js

Node.js – это серверная технология, основанная на JavaScript-движке V8 компании Google. Данная система обладает хорошими средствами маршрутизации и поддерживает не программные потоки или отдельные процессы, а также асинхронный ввод-вывод, который управляется событиями. Технология подходит для веб-приложений, не выполняющих сложных вычислений, но к которым происходит частое обращение.

Для работы платформы Node используется виртуальная машина V8, благодаря которой производительность Node значительно возрастает, поскольку устраняются промежуточные этапы создания исполняемого кода, поскольку происходит компиляция непосредственно в машинный код. В связи с тем, что Node применяет JavaScript на стороне сервера, появляются следующие преимущества:

- Разработчики могут создавать веб-приложения на одном языке, благодаря чему снижается потребность в переключении контекста при разработке серверов и клиентов. При этом обеспечивается совместное использование кода клиентом и сервером, например кода проверки данных, вводимых в форму, или кода игровой логики.

- Популярнейший формат обмена данными JSON является собственным форматом JavaScript.

- Язык JavaScript применяется в различных базах данных NoSQL (например, в CouchDB и MongoDB), поэтому подключение к таким базам данных осуществляется в естественной форме. Например, оболочкой и языком запросов для базы данных MongoDB является язык JavaScript; языком проецирования/сведения для базы данных CouchDB также является JavaScript.

- Целью компиляции в Node.js является JavaScript, к тому же в настоящее время существует ряд других языков программирования, компилируемых в JavaScript.

- В Node используется единственная виртуальная машина (V8), совместимая со стандартом ECMAScript 2015. Другими словами, вам не придется ожидать, пока во всех браузерах станут доступны все новые средства языка JavaScript, связанные с платформой Node.

Асинхронность событий и однопоточность приложений

При обращении к веб-приложению Node не создает новый программный поток или процесс для каждого из запросов, а слушает прописанные события. Если произошло то или иное событие, то веб-сервер Node выполняет прописанный для него сценарий. Веб-сервер Node.js обрабатывает события по принципу FIFO (First-In-First-Out), причем события выполняются асинхронно, что позволяет избежать блокирования запросов к веб-серверу. Также следует отметить, что веб-приложение, написанное на Node.js является однопоточным, т.е. использует один поток для обработки всех запросов. Именно асинхронная природа Node.js делает возможным обработку большого числа взаимодействий на веб-странице и позволяет создать отзывчивый интерфейс, в котором отсутствует потребность в ожидании подключения к ресурсам сервера.

Особенность разработки веб-приложений

Node.js поддерживает HTTP-интерфейс при помощи модуля http. Чтобы создать HTTP-сервер, требуется вызвать функцию http.createServer(). Эта функция имеет единственный аргумент – функцию обратного вызова, которая вызывается в каждом HTTP-запросе, принимаемом сервером.

Для каждого полученного сервером HTTP-запроса вызывается функция обратного вызова. Еще до выполнения обратного вызова Node выполняет синтаксический разбор запроса вплоть до конца HTTP-заголовков. Node не может начать разбор тела запроса до тех пор, пока не будет сделан обратный вызов. Это поведение отличается от поведения серверных сред разработки, таких как PHP, в которых синтаксический разбор заголовков и тела запроса

происходит до начала выполнения кода приложения. Node предлагает низкоуровневый интерфейс, позволяющий обрабатывать данные тела запроса по мере их разбора.

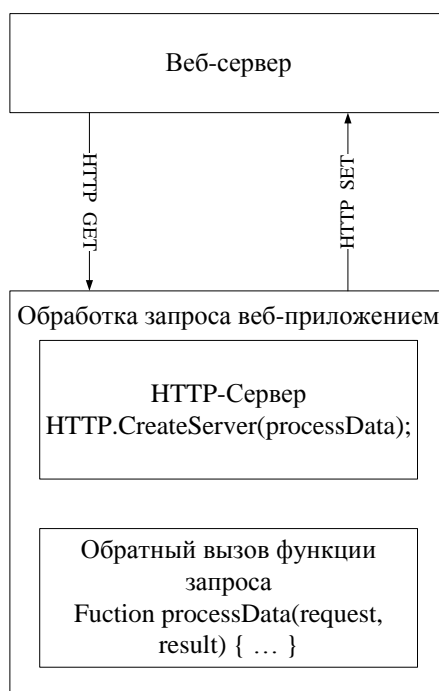


Рисунок 1. Жизненный цикл HTTP-запроса в Node.js с использованием HTTP-сервера.

На рисунке 1 изображен жизненный цикл HTTP-запроса. HTTP-клиент (веб-браузер) делает HTTP-запрос к веб-приложению. Node-приложение принимает подключение и данные запроса. HTTP-сервер разбирает запрос до конца, затем вызывает функцию обратного вызова (`processData`). Функция обратного вызова выполняет прописанный в ней код и возвращает отклик. Далее запрос возвращается обратно в браузер, формирующим подходящий HTTP-отклик для клиента.

RESTful интерфейс

Node.js поддерживает API-интерфейс REST (Representational State Transfer – передача состояний представления). Данную концепцию предложил Рой Филдинг в 2000 году, являющийся одним из соавторов спецификаций протоколов HTTP 1.0 и 1.1. В соответствии с данной концепцией установлены HTTP глаголы, такие как GET, POST, PUT, DELETE. Чтобы создать REST-

совместимый сервер необходимо реализовать эти HTTP глаголы. Каждый глагол выполняет строго определенную функцию:

GET – получение списка запрашиваемых элементов;

POST – добавление элемента в ресурс;

DELETE – удаление элемента из ресурса;

PUT – обновление существующего элемента.

Ниже приведен программный код, реализующий обработку

```
var http = require('http')
var server = http.createServer(function(req, res){
  switch (req.method){
    case 'GET':
      //извлечение элементов из массива
      items.forEach(function(item,i){
        //действия над извлеченными элементами
        res.end();});
      break;
    case 'POST':
      //перемещение фрагментов в массив
      var item = '';
      req.on('data', function(chunk){
        item+=chunk; });
      req.on('end', function(){
        items.push(item);
        res.end();});
      break;
    case 'DELETE':
      var urlPath = url.parse(req.url).pathname;
      var i = parseInt(urlPath.slice(1),10);
      if(isNaN(i)){
        //проверка корректности числа
        res.statusCode = 400;
        res.end('Invalid item id');}
      else if(!items[i]){
        // проверка существования элемента в массиве по запрашиваемым
индексом
        res.statusCode = 404;
        res.end('Item not found');}
      else{
        //Удаление запрошенного элемента
        items.splice(i,1);
        res.end('OK\n');}
      break; }
    case 'PUT':
      var urlPath = url.parse(req.url).pathname;
      var i = parseInt(urlPath.slice(1),10);
      if(isNaN(i)){
        //является ли i числом
        res.statusCode = 400;
        res.end('Invalid item id');}
      else if(!items[i]){
        // проверка существования элемента в массиве по запрашиваемым
индексом
        res.statusCode = 404;
        res.end('Item not found');}
      else{
        //Редактирование запрошенного элемента
        items[i].name = req.body.name;
```



```
        items[i].description = req.body.description;
        res.end('OK\n');}
break; } }
```

В Node.js проверка использованного HTTP-метода осуществляется посредством свойства `req.method`. Определив метод, прописывается его поведение. Когда система синтаксического разбора HTTP-кода в Node считывает и разбирает данные запроса, они становятся доступны в формате `data` и содержат фрагменты разобранных данных, готовых для дальнейшей обработки. Событие `req.end` обозначает конец запроса. Поле `statusCode` объекта `res` позволяет обработать ошибку сервера, присвоив этому полю определенное значение.

MVC-шаблон проектирования веб-приложения в Node.js

Node.js позволяет реализовать архитектуру MVC (Model-View-Controller). Данная архитектура стала популярной в связи с возрастающей популярностью веб-приложений. Она позволяет создать структурированный код, разграничивая функционал приложения.

Модель представляет данные и методы работы с данными, изменяет свое состояние в соответствии с запросами. Модель не содержит информацию о том, как данные попадут к пользователю. Представление отвечает за отображение информации, т.е. за визуализацию данных. Контроллер обеспечивает связь между пользователем и серверной частью веб-приложения: контролирует ввод данных, управляет потоками данных, преобразовывает модель. Важно отметить, что модель независима от реализации контроллера и представления, т.е. модель строится независимо от визуализации [1].

Node-приложения, реализующие архитектуру MVC, могут разграничить уровень модели и представления, т.е. серверную и клиентскую часть веб-приложения. Так, модель и контроллер будут функционировать на сервере, а представление – на клиентской стороне, т.е. в веб-браузере.

Тестирование Node-приложений

По мере расширения функционала веб-приложения возрастает риск внедрения в него ошибок. Большую популярность набирает

автоматизированное тестирование, при котором логика проверки функционального блока описывается программистом. Существует два основных вида автоматизированного тестирования: модульное и приемочное тестирование.

Модульное тестирование непосредственно проверяет логику и функционал приложения. Данный вид тестирования позволяет проверить отдельные части приложения и избежать ошибок на ранних этапах разработки веб-приложения. При использовании модульного тестирования есть гарантия, что текущие изменения не приведут к возникновению ошибок. Приемочное тестирование применяется при отладке приложений и подразумевает контроль сценариев, поступающих из браузера, и проверку функциональности веб-приложения.

1.1.2 Технология Ruby on Rails

Со времен своего дебюта в 2004 году, Ruby on Rails быстро стал одним из самых мощных и популярных фреймворков для построения динамических веб-приложений. Ruby on Rails на сто процентов открыт, доступен в силу MIT License, и, как результат, его можно загружать и использовать бесплатно. Rails также обязан своим успехом своему изящному и компактному дизайну; используя податливость лежащего в его основе языка Ruby, Rails фактически создает предметно-ориентированный язык (domain-specific language) для написания веб-приложений. В результате много общих задач веб-программирования – таких как генерирование HTML, создание моделей данных и маршрутизация URI – легки с Rails, а результирующий код программ краток и читаем.

Rails-приложения пишутся на Ruby — современном объектно-ориентированном языке сценариев. Данный язык программирования позволяет создать лаконичный и легко читаемый программный код.

Все Rails-приложения выполняются с использованием архитектуры Модель-Представление-Контроллер (Model-View-Controller, MVC) [3, с. 12].

Роль модели в Ruby on Rails является такой же, как и в других технологиях, реализующей архитектуру MVC – модель является контейнером для работы

с базой данных. В данной технологии таблицы, описанные в базе данных, являются моделями в приложении, а поля таблицы – полями модели соответственно. Однако такой подход затрудняет сочетаемость реляционных баз данных с объектно-ориентированными языками программирования, поскольку объекты описываются данными и операциями, а базы данных – наборами значений. Операции, легко выражаемые в реляционных понятиях, иногда сложно запрограммировать в объектно-ориентированных системах. Также справедливо и обратное утверждение.

Для осуществления связи модели с базой данных применяется объектно-реляционное отображение данных. Так, если в базе данных есть таблица Book, то в проекте Ruby on Rails будет класс Book. Строки в этой таблице соответствуют объектам класса.

Active Record — это ORM-вставка, предоставляемая Rails. Она строго следует стандартам ORM-модели: таблицы отображаются на классы, строки — на объекты, а столбцы — на свойства объекта. От большинства других ORM-библиотек она отличается способом конфигурирования. Основываясь на соглашении и приступая к работе с оптимальными настройками по умолчанию, Active Record сводит к минимуму объем настроек, выполняемых разработчиками [3, с.52].

```
require 'active_record'
class Book < ActiveRecord::Base
end
book = Book.find(1)
book.name = "Война и мир"
book.save
```

В данном коде извлекается книга с идентификационным номером 1, изменяется ее название и сохраняются изменения.

Представление и контроллер в архитектуре MVC тесно связаны. Именно поэтому разработчики Rails приняли решение об их объединении в единый компонент Action Pack. Однако, при таком объединении сохраняется четкое разделение логики.

Представление в Rails отвечает за создание полного или частичного ответа, отображаемого в браузере, обработанного приложением или посланного в

виде электронной почты. В простейшем виде представление является фрагментом HTML-кода.

Динамическое содержимое в Rails генерируется шаблонами трех видов. В самой распространенной схеме создания шаблонов, которая называется встроенным Ruby — Embedded Ruby (ERb), фрагменты кода Ruby вставляются в представляемый документ, что во многом похоже на способ, применяемый в других веб-средах, например в PHP или в JSP. При всей гибкости данного подхода некоторые специалисты озабочены тем, что он нарушает сам смысл MVC. Вставляя код в представление, мы рискуем заложить в него логику, которая должна быть в модели или в контроллере [3, с.53].

ERb можно также использовать для конструирования на сервере JavaScript-фрагментов, выполняемых в браузере. Эта технология отлично подходит для создания динамичных AJAX-интерфейсов.

В Rails также имеется такой инструмент, как XML Builder, позволяющий конструировать XML-документы, использующие код Ruby, — структура генерируемого XML будет автоматически следовать за структурой кода.

Контроллер Rails является логическим центром приложения. Он координирует взаимодействие между пользователем, представлениями и моделью. Контроллер отвечает за выполнение нескольких важных функций:

- перенаправление внешних запросов внутренним действиям;
- управление кэшированием, что ускоряет работу приложения в несколько раз;
- управление вспомогательными модулями, расширяющими возможности шаблонов представлений без увеличения объемов кода;
- управление сессиями, дающими пользователю ощущение непрерывного взаимодействия с приложением.

Многие считают Ruby on Rails наиболее продуманной технологией разработки программного обеспечения, поскольку она уже содержит в себе все необходимые модули. Однако в рамках данной работы необходимо предусмотреть

реть возможность подключения к различным источникам данных, что не предусмотрено данной технологией.

1.1.3 Технология ASP.NET

В конце 1990-х годов появилась платформа ASP.NET. Она стала продолжателем таких технологий, как ASP (Active Server Pages) и JSP (Java Servlet Pages). ASP.NET – это набор технологий для разработки веб-приложений, работающих под управлением веб-сервера IIS (Internet Information Services).

ASP.NET Web Forms

ASP.NET Web Forms – технология разработки веб-приложений на основе объектно-ориентированного подхода с поддержкой шаблонов на основе форм. Файлы веб-форм являются контейнерами серверных элементов управления, содержащихся в пространстве имен System.Web, имеющие расширение .aspx. Также в составе проекта веб-форм есть файлы, содержащие бизнес-логику приложений.

Web-формы схожи с Windows-формами, поскольку серверные элементы управления могут отображать данные и инициировать события, на которые могут быть привязаны обработчики.

Модель ASP.NET Web Forms изначально проектировалась для того, чтобы реализовать потенциал RAD (Rapid Application Development) в среде Web. Таким образом, главным определяющим фактором для большинства основных характеристик и ключевых концепций ASP.NET было стремление к производительности программирования. Модель Web Forms базируется на трех основных концепциях: обратной передаче страниц, состоянии просмотра и серверных элементах управления. Каждый запрос HTTP, передаваемый веб-серверу и сопоставляемый со средой выполнения ASP.NET, проходит несколько стадий, в которых центральное место занимает обработка события обратной передачи (postback). Событие обратной передачи — главное действие, которое ожидает получить пользователь в результате обработки своего запроса [4, с. 3].

Сначала происходит обработка запроса для извлечения подготовительной информации, необходимой для успешного действия обратной передачи. В эту информацию входит состояние элементов управления, совместно формирующих итоговую разметку HTML-страницы. После обратной передачи для браузера генерируется ответ HTML с новым состоянием элементов, которое будет использоваться при следующем запросе.

Страница ASP.NET базируется на одном компоненте формы, содержащем все элементы ввода, с которыми может взаимодействовать пользователь. Форма также содержит элементы отправки данных — например, кнопки или ссылки.

При отправке данных формой содержимое текущей формы отправляется на серверный URL-адрес — по умолчанию совпадающий с URL-адресом текущей страницы. Действие по отправке содержимого той же странице называется обратной передачей (postback). В ASP.NET страница отправляет все содержимое своей уникальной формы самой себе. Иначе говоря, страница представляет собой структурный блок приложения, содержащий как визуальный интерфейс, так и логику обработки действий пользователя.

Так, например, при нажатии на кнопку Button1 срабатывает событие Button1_Click. При этом в коде обработчика программист может обновить пользовательский интерфейс, изменяя состояние серверных элементов управления:

```
public void Button1_Click(object sender, EventArgs args)
{
    // В элементе Label выводится содержимое текстового поля
    Label1.Text = "The textbox contains: " + TextBox1.Text;
}
```

Состояние просмотра (View State) представляет собой словарь, используемый страницами ASP.NET для хранения состояния своих дочерних элементов управления между обратными передачами. Состояние просмотра играет ключевую роль в реализации модели обратной передачи. Без него отслеживание состояния в ASP.NET было бы невозможно. View State - это уникальное (и зашифрованное) скрытое поле, в котором хранится словарь значений всех элементов управления (уникальной) формы страницы ASP.NET. По умолчанию

каждый элемент страницы сохраняет свое полное состояние — включая значения всех его свойств — в состоянии просмотра. Эти данные передаются клиенту и принимаются сервером с каждым запросом страницы. Тем не менее они никогда не используются на стороне клиента.

Серверные элементы управления играют ключевую роль в модели ASP.NET Web Forms. Вывод страницы ASP.NET определяется в виде комбинации литералов HTML и разметки серверных элементов управления ASP.NET. Серверный элемент управления представляет собой компонент с открытым интерфейсом, который может настраиваться с использованием тегов разметки, дочерних тегов и атрибутов. Каждый серверный элемент управления имеет уникальный идентификатор, который однозначно определяет его.

В разметке страницы ASP.NET серверные элементы управления отличаются от простых строковых литералов HTML по наличию атрибута `runat`. Все теги, не имеющие атрибута `runat`, интерпретируются как литералы HTML и передаются в выходной поток ответа без дополнительной обработки. Все, что имеет пометку `runat`, идентифицируется как серверный элемент управления. Серверные элементы управления изолируют пользователя от фактического генерирования кода HTML и JavaScript. Программирование серверного элемента управления сводится к заданию свойств компонента, предназначенного для многократного использования. Однако при обработке серверный элемент управления генерирует код HTML.

На первых порах существования ASP.NET Web Forms ограниченность контакта с HTML и JavaScript была безусловным «плюсом». Однако стремительный рост популярности AJAX в середине прошлого десятилетия изменил точку зрения на возможности веб-приложений, и что еще важнее, значительно изменил ожидания пользователей. В результате от веб-приложений требуется значительно более высокая интерактивность и скорость реагирования на действия пользователя. Один из способов ускорения реагирования — увеличение объема сценарного кода, который выполняется в браузере только при отобра-

жении конкретной страницы. Из-за этого простого факта разработчикам потребовалась более высокая степень контроля за генерируемой разметкой.

ASP.NET MVC

Компания Microsoft представила ASP.NET MVC как альтернативу Web Forms, однако MVC обладает рядом преимуществ. Разделение веб-приложения на модель, контроллер и представление позволяет улучшить модульное тестирование, вследствие чего повышается контроль над веб-приложением. Главной проблемой веб-форм является передача большого объема данных в объекте состояния, что исправлено в MVC-приложениях.

Как и в других технологиях, в основе MVC-архитектуры приложения лежат модели, представления и контроллеры. Модель выступает в качестве контейнера данных. Контроллер обрабатывает запросы клиента и возвращает результат этих запросов. Представление отображает данные модели.

MVC-проект имеет следующую структуру: папки Models, Views, Controllers содержат классы моделей, представлений и контроллеров соответственно; папка Scripts содержит клиентские JavaScript-файлы; файл Default.aspx содержит обработчики событий по загрузке начальной страницы; файл Global.asax содержит таблицу маршрутизации, соотносящую запросы к веб-приложению с методами контроллеров приложения; файл Web.config содержит описание конфигурации веб-приложения, в том числе строку подключения к базе данных.

Жизненный цикл запроса ASP.NET MVC приложения очень прост. Клиент (веб-браузер) посылает HTTP-запрос, который попадает на обработку в таблицу маршрутизации. Именно она определяет наименование контроллера и действие в нем, который должен обработать поступивший запрос. На основании параметров запроса метод (действие) контроллера выполняет некоторый алгоритм, генерирует представление для отображения пользователю.

Основным связующим звеном между контроллером и представлением является коллекция View Data, позволяющая осуществлять обмен данными между этими компонентами.

После генерации разметки движком представления, веб-сервер передает ее в качестве ответа по протоколу HTTP. Это является последним этапом жизненного цикла запроса в MVC-приложении.

Модель веб-приложения представляет собой простые классы, описывающих структуру данных. Например, модель книги имеет следующий вид:

```
public class Book
{
    // ID книги
    public int Id { get; set; }
    // наименование
    public string Name { get; set; }
    // автор
    public string Author { get; set; }
    // цена
    public int Price { get; set; }
}
```

Данные моделей, как правило, хранятся в базе данных. Для осуществления связи между моделью и базой данных применяется фреймворк Entity Framework , позволяющий избежать написания sql-запросов и придерживаться объектно-ориентированного подхода на протяжении всей работы над веб-приложением [5].

Для осуществления подключения к базе данных через Entity Framework создается класс контекста данных (BookContext), производный от класса DbContext. Данный класс содержит одно или несколько свойств типа DbSet<T>, где T – тип модели, хранящейся в базе данных.

```
public class BookContext : DbContext
{
    public DbSet<Book> Books { get; set; }
}
```

Существует три подхода, позволяющих использовать EntityFramework: Code-First, Model-First и Database-First. Подход Model-First применяется в том случае, когда в приложении уже есть модель данных, на основе которой будет создана структура базы данных. При подходе Database-First на основе суще-

ствующей базы данных создается модель в MVC-приложении. Подход Code-First совместил функции предыдущих двух подходов [6].

Применение подхода Code-First продемонстрировано в методе Edit контроллера BookController.

```
public class BookController: Controller{
    public ActionResult Edit(int id) {
        BookContext db = new BookContext ();
        var book = db.Books.Where (b => b.Id == id).Select(b => b).Single ();
        return View(book);
    }
    [HttpPost]
    public ActionResult Edit(int id, FormCollection collection) {
        try {
            BookContext db = new BookContext ();
            var book = db.Books.Where (b => b.Id == id).Select(b => b).Single ();
            UpdateModel(book);
            db.SaveChanges();
            return RedirectToAction("Index"); }
        catch { return View(); } }
}
```

Первый метод данного контроллера отображает данные о запрашиваемой сущности книги. По завершению редактирования данных о книге пользователь сохраняет их. В этом случае производится переход в метод контроллера с Post-идентификатором. Данные отредактированной сущности книги сохраняются в базе данных.

Маршрутизация

Платформа MVC преобразует URL-запросы в контроллеры, используя маршрутизацию ASP.NET [4, с. 10]. Файл Global.asax включает в себя метод RegisterRoutes, который содержит следующий оператор:

```
routes.MapRoute( "Default", "{controller}/{action}/{id}", new { controller = "Home", action = "Index", id = UrlParameter.Optional } );
```

Этот оператор вызывает метод MapRoute, чтобы зарегистрировать маршрут, который решает для нашего приложения две важные задачи. Во-первых, он устанавливает формат для запросов так, чтобы после визуализации представлений обратные ссылки на приложение MVC были представлены в форме контроллер/действие/идентификатор (" {controller}/{action}/{id} "). Например, если бы требовалось просмотреть сведения о книге с ID, равным 7, то URL-адрес должен был бы иметь вид /book/details/7. Изменение формата URL-адресов для приложения MVC делается также в этом методе.

Второй важной частью оператора является последний аргумент метода `MapRoute`, указывающий значения по умолчанию, которые будут использоваться, когда они не переданы как часть URL-адреса. Именно поэтому метод `Index` контроллера вызывается при запросе `/book` — действие не задано, поэтому используется значение `Index`, принятое по умолчанию. Указание `id = UrlParameter.Optional` означает, что идентификатор записи добавлять не нужно, если он не передан в запросе.

1.2 Обзор конкурентных систем

На данный момент не известно об аналогах систем автоматического формирования отчетов в России, однако существует две больших системы: «WinCC/DataMonitor» компании Siemens (Германия) и «Ampla» Schneider Electric (Франция). Главным преимуществом веб-портала «Smart Reports» перед данными конкурентами является стоимость и доступность поддержки программного продукта, а также его ориентированность на особенности российского рынка.

Рассмотрим подробнее особенности функционала конкурентов.

«WinCC/DataMonitor» является частью WinCC Plant Intelligence и применяется для отображения и анализа текущего состояния процесса и исторической информации на офисном ПК при помощи стандартного инструментария, такого как Microsoft Internet Explorer или Microsoft Excel. Данная система состоит из четырех подсистем:

- Process Screens – просмотр визуализации производственного процесса при помощи Internet Explorer.
- Trends&Alarms – отображение и анализ архивных данных производственного процесса, экспорт данных из Internet Explorer в MS Excel.
- Excel Workbooks – разработка отчетов и анализ данных в MS Excel. Публикация отчетов в WebCenter.
- Reports – запуск формирования отчетов по событию или по расписанию в форматах Excel или PDF.
- WebCenter – построение Internet портала, как центральной точки доступа ко всем данным WinCC.

Исторические данные предоставляются веб-сервером в режиме реального времени. Обработка этих данных производится при помощи заранее подготовленных шаблонов. Также есть возможность динамической компоновки данных в соответствии с возникающими требованиями.

Другим аналогом является разработка компании Schneider Electric «Ampla». Данный программный продукт ориентирован на горнодобывающую

отрасль. Инструментарий данной системы охватывает 11 показателей производства: время простоя оборудования, производительность, метрики, инвентаризация, энергопотребление, качество, планирование, сопровождение, знания, доходы, стоимость. Основываясь на этих показателях «Ampla» позволяет производить операции над историческими данными. Следует отметить, что разработчики этой платформы стремились к созданию отзывчивого и интуитивно понятного графического интерфейса при визуализации данных.

Таким образом, веб-портал «Smart Reports» не привязан к конкретной области производства, как это реализовано в «Ampla». Среди недостатков вышеперечисленных программных платформ можно выделить следующее:

- 1) высокая стоимость программных продуктов;
- 2) зарубежный производитель не учитывает особенности Российского производства;
- 3) сложности в конфигурации таких систем;
- 4) недостаточно развитая техническая поддержка в России, что влияет на качество ее эксплуатации;
- 5) закрытый исходный код;

Выводы по главе

В результате приведенного обзора литературы технологий разработки веб-приложений можно сделать вывод об актуальности использования ASP.NET MVC Framework для реализации проекта «SmartReports». В силу различий между ASP.NET MVC и Ruby on Rails разработчики, ранее работавшие на платформе .NET с легкостью освоят ASP.NET MVC платформу, в то время как разработчикам, предпочитавшим языки программирования Python или Ruby, будет гораздо легче начать работать с Ruby on Rails. Если Ruby on Rails имеет полный стек технологий для работы с базами, то у .NET-платформы есть огромный выбор реализации функционала посредством различных технологий.

Работая с ASP.NET MVC, разработчик вправе самостоятельно выбирать компоненты для построения приложения, но они уже не будут так тесно связаны, как их эквиваленты в Rails.

При сравнении ASP.NET и node.js можно прийти к выводу, что выбор той или иной платформы зависит от предпочтений разработчика, поскольку в обоих случаях есть возможность использовать сторонние библиотеки, обе платформы обладают достаточной производительностью, реализуют асинхронную обработку запросов, а также MVC-архитектуру. Таким образом, для разработки корпоративного веб-портала «SmartReports» была выбрана технология ASP.NET MVC.

В результате обзора систем-аналогов можно сделать вывод о том, что разрабатываемая система может получить широкое применение на российском рынке в связи с отсутствием недостатков, имеющих у зарубежных аналогов систем.

Глава 2. Постановка и решение задачи

1.1 Описание задачи динамического формирования отчетов

Отчет – это множество данных предприятия, отображаемые в некотором графическом представлении. Данные в отчете могут быть отображены следующим образом:

- таблица
- график
- диаграмма
- гистограмма

Построение отчета происходит в несколько этапов:

1. Выборка и подготовка данных
2. Создание шаблона отчета.
3. Заполнение шаблона отчета обработанными данными.

Рассмотрим последовательно операции по составлению алгоритма обработки данных.

Алгоритм обработки данных

Построение отчета предполагает наличие данных, на основе которых он будет сформирован. Данные о деятельности предприятия хранятся в различных хранилищах данных. Сперва производится описание подключения к источникам данных. Затем, используя эти подключения, выбираются данные для обработки и последующего отображения в отчете. Зачастую выбранные из источников данные избыточны и неинформативны. В связи с этим требуется произвести ряд операций над данными.

Выбранные из источников данные обрабатываются в соответствии с установленным пользователем веб-портала алгоритмом и поступают в подготовленный шаблон отчета.

Алгоритм построения отчета

Перед заполнением отчета данными подготавливается форма отчета. Форма отчета – одна или несколько графических форм отображения данных в отчете, такие как таблица, график, диаграмма и т.д. Подготовленный шаблон отчета заполняется обработанными данными. Графическое представление данных должно быть информативно для пользователя.

1.2 Анализ методов формирования отчетной документации

Алгоритм формирования отчетных документов может быть реализован двумя способами:

- вариант №1– формирование по запросам пользователя;
- вариант №2– динамическое формирование алгоритмов обработки данных и шаблонов отчетов.

В данной главе производится анализ двух методов формирования отчетной документации и обосновывается выбранный для реализации подход.

2.2.1 Метод формирования отчетных документов посредством разработки отчетов по запросам пользователей

В данном подходе участвуют две роли: пользователь и разработчик.

Метод, основанный на разработке алгоритма отчетов программистом на основании предоставленных требований, позволяет пользователю просматривать готовые отчеты. Непосредственную разработку алгоритмов формирования отчетов производит разработчик (программист): создается набор классов по обработке исходных данных и формированию файла отчета. На рисунке 2 изображена архитектура веб-приложения при втором варианте исполнения системы.

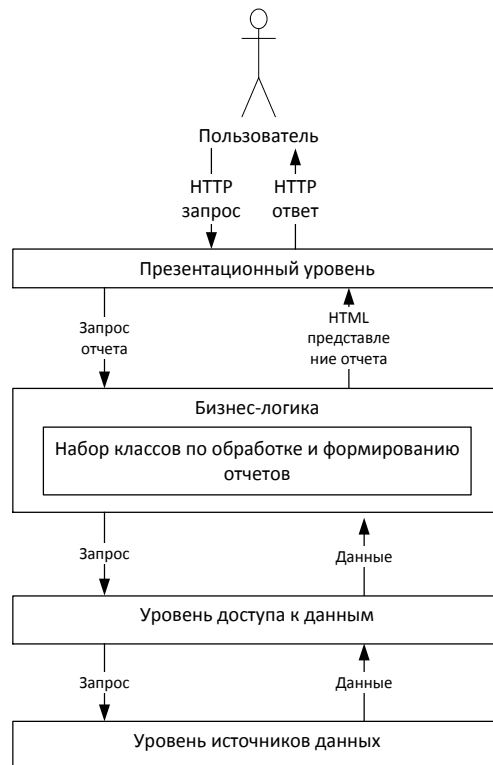
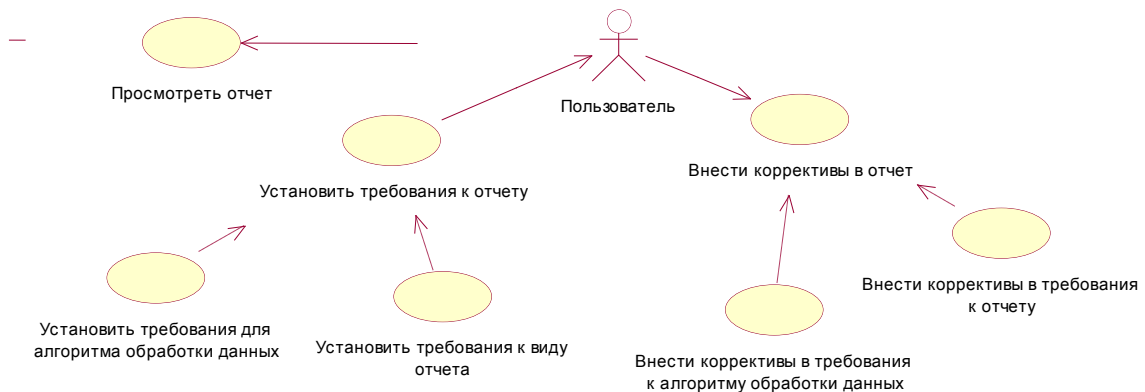


Рисунок 2. Архитектура варианта системы при реализации метода №2.

Исходя из такой архитектуры, пользователь может выбрать только готовую реализацию отчета, но не имеет возможности внести в него изменения. Пользователь запрашивает доступный отчет на веб-портале, затем написанные разработчиком классы извлекают и обрабатывают данные по прописанному в них алгоритму и формируют отчет.

На рисунке 3 продемонстрирована диаграмма вариантов использования при методе формирования отчетных документов №2 для роли конечного пользователя.



Рисунков 3. Диаграмма вариантов использования варианта системы при реализации метода №2 для роли конечного пользователя.

Для создания отчета конечному пользователю необходимо предоставить список требований к отчету: какие данные должны отображаться в отчете, в каком виде должны быть представлены эти данные, параметры для обработки данных (временные рамки, эталонные значения и т.д.). При возникновении потребности во внесении изменений в отчет необходимо составить список корректировок к отчету. Все эти требования являются входными данными для разработчика (программиста) при разработке или корректировке алгоритма обработки данных или алгоритма формирования отчета. На рисунке 4 продемонстрированы варианты использования варианта системы при реализации метода №2 для роли разработчика отчетов.

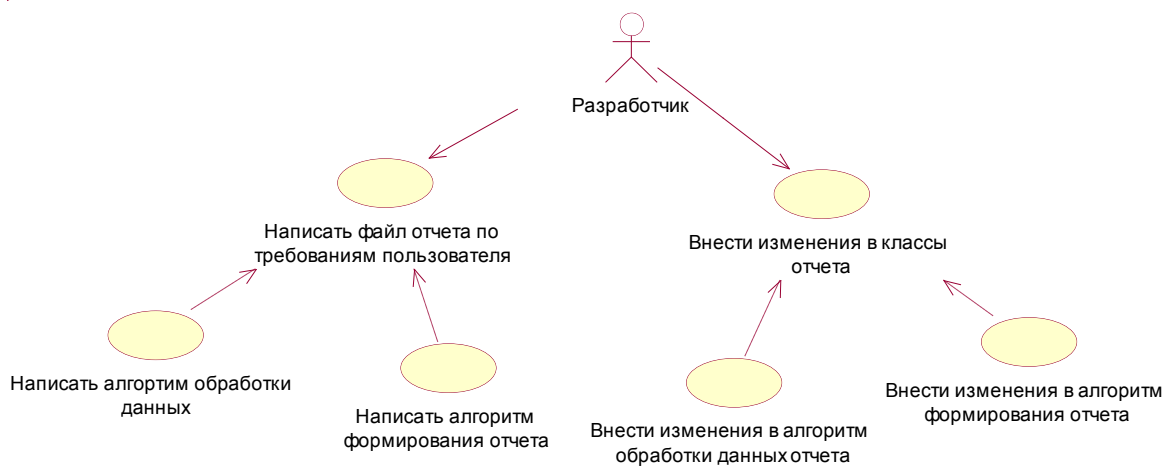


Рисунок 4. Диаграмма вариантов использования варианта системы при реализации метода №2 для роли разработчика отчетов.

Программист разрабатывает как новые алгоритмы по обработке данных и формированию отчетов, так и редактирует уже существующие алгоритмы на основе составленного пользователем списка требований к отчету.

На рисунке 5 представлена диаграмма последовательности формирования алгоритма отчета разработчиком на основании требований пользователя. В данном варианте исполнения системы пользователь не принимает прямого участия в формировании структуры отчета и алгоритмов по обработке данных. Также можно заметить, что процесс непосредственного формирования алго-

ритмов осуществляется без привлечения средств веб-портала. Разработчик создает файл отчета в среде разработки программного обеспечения, например Microsoft Visual Studio.

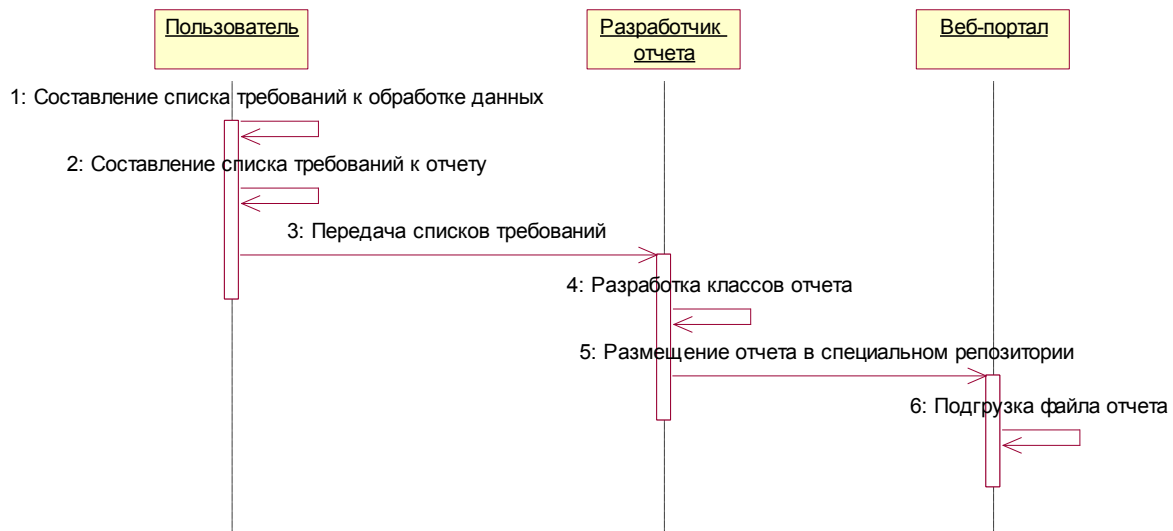


Рисунок 5. Диаграмма последовательности формирования алгоритма отчета при втором варианте исполнения системы.

В рамках данного метода формирования отчетных документов задачей пользователя является лишь составление списка требований к отчету: определить данные для обработки, критерии обработки и вид их представления. Пользователь увидит только конечный результат – просмотрит готовый файл отчета.

При таком подходе пользователь может обладать минимальным знанием о системе хранения производственных данных, но должен давать четкие формулировки требований к отчету для достижения желаемого результата. Также при прекращении сопровождения разработанного программистом отчета могут возникнуть сложности при внесении изменений в него, так как потребует время для ознакомления с программным кодом.

2.2.2 Метод обработки данных с динамическим формированием отчетов конечными пользователями

В рамках метода с динамическим формированием отчетов выделено две роли: пользователь и администратор.

На рисунке 6 продемонстрирована архитектура многоуровневой системы обработки данных и формирования отчетов.

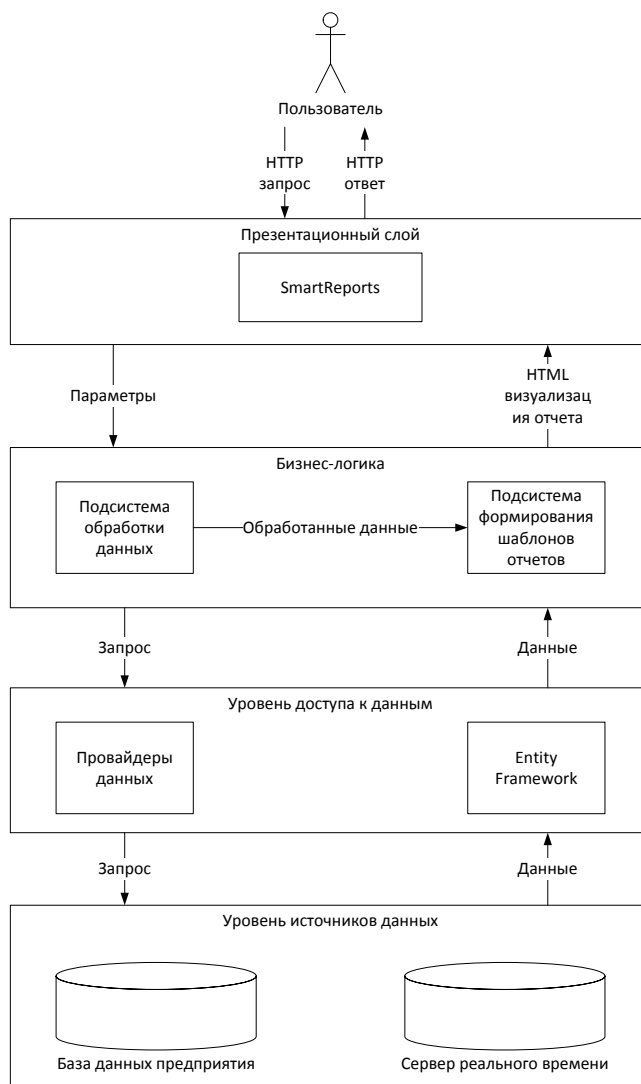


Рисунок 6. Архитектура варианта системы при реализации метода формирования отчетов №2.

Посредством презентационного слоя веб-приложения (View) пользователь задает параметры алгоритма обработки данных и шаблонов отчетов. Далее в соответствии с установленными пользователем требованиями подсистема обработки данных осуществляет выборку данных посредством различных про-

вайдеров баз данных и технологий доступа к данным. Обработанные данные являются основой для отчета, поэтому подаются на вход подсистеме формирования шаблонов отчетов. По сравнению с вариантом системы №1, такая архитектура позволит пользователю самостоятельно вносить изменения в алгоритмы, минуя такие промежуточные шаги, как составление списка требований к отчету, отправка запроса разработчику и т.д. При внедрении данной архитектуры есть возможность настроить дополнительные источники данных, индивидуальные для конкретного предприятия.

Реализованная с применением такого подхода система обладает широким пользовательским функционалом. Круг возможностей конечного пользователя гораздо шире по сравнению с методом №1. Диаграмма вариантов использования приведена на рисунке 7.

Конечному пользователю доступно две основные функции:

- конфигурирование алгоритма обработки данных
- настройка параметров построения отчета.

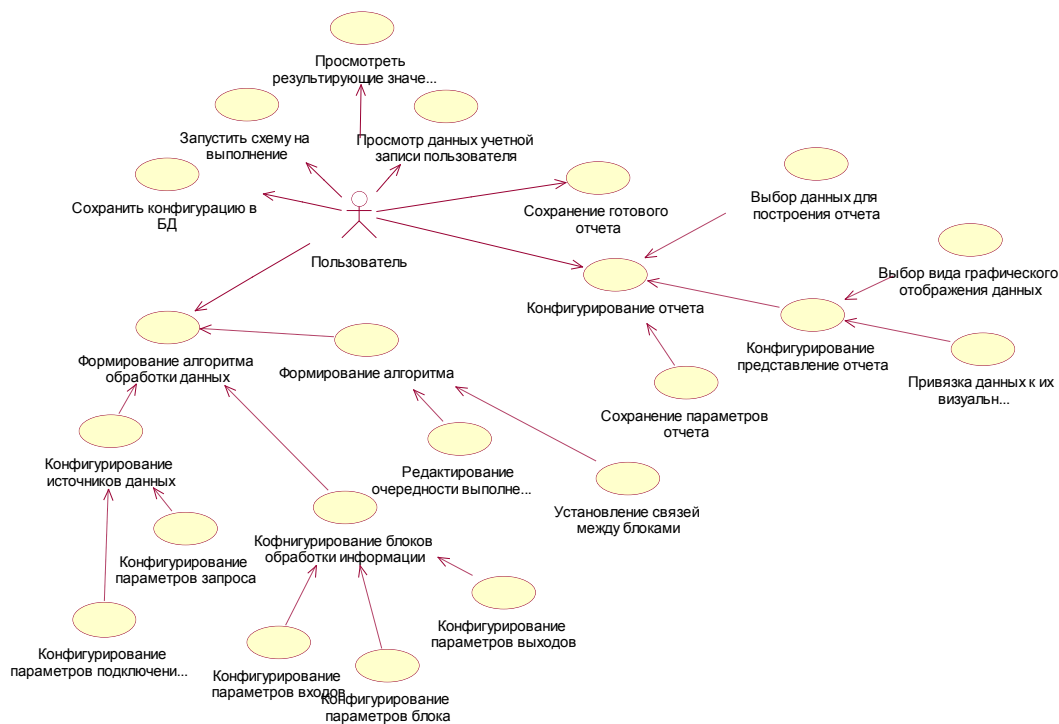


Рисунок 7. Диаграмма вариантов использования веб-портала для роли конечного пользователя.

Перед формированием отчета требуется обработать «сырые», избыточные данные в соответствии с определенным алгоритмом: сделать выборку по параметрам, преобразовать величины, произвести расчетные операции и т.п. Во-первых, для обработки каких-либо данных их следует получить. Это возможно благодаря настройке подключения к источнику данных: при выборе определенного типа источника данных заполняются параметры, необходимые для подключения к нему. Все классы источников данных являются наследниками класса `IDataSource` (приложение Г). Если данные компании хранятся в уникальной базе данных, то возможно реализовать еще один класс источников данных, прописав алгоритм подключения в методе `Connect`.

Алгоритм обработки данных строится из блоков. Блоки делятся на два типа: запросы и операции. Блоки-запросы делают выборку из источников данных, блоки операций обрабатывают выбранные данные. Каждый блок является классом-наследником интерфейса `IBlock` (приложение Б). Каждый блок имеет входы и выходы. На вход поступает информация для обработки (для блоков операций) или параметры запросов (для блоков запросов). Целостность алгоритма обеспечивается связями между блоками. Выход одного блока соединяется со входом другого блока. Посредством связи организуется не только формирование последовательности выполнения блоков в алгоритме, но также и передача данных с выхода одного блока на вход другого. Таким образом, возможно реализовать следующую цепочку: выбранные из хранилища данные поступают на вход в блок-обработчик, подвергаются некоторой обработке, а на выходе данного блока-обработчика получаем уже преобразованные данные.

Далее делается запрос к этому источнику данных, на выходе которого получаем первичные данные. Запросы могут быть следующих типов: `Sql`, `OPC DA`, `OPC HDA`. Каждый тип запроса представлен классом-наследником интерфейса `IBlock` (приложение Б), поскольку участвует в непосредственном построении алгоритма. Для каждого блока запроса необходимо определить, через какой источник данных производить выборку. После выбора источника данных тре-

буется написать запрос к нему. В зависимости от потребностей пользователь может подвергнуть эти данные обработке, либо оставить как есть, и воспользоваться ими при конфигурировании параметров отчета. Если же эти данные неинформативны, пользователь имеет возможность подвергнуть их вычислительным операциям.

При настройке блоков вычислительных операций, у которых предусмотрена пользовательская настройка входов и/или выходов, возможно выбрать тип входа или выхода, т.е. определить, в каком формате данные поступят на обработку и в каком формате они будут в результате этой обработки. Такая динамическая конфигурация блоков операций позволяет обработать одни и те же данные несколькими подходами или несколькими алгоритмами, получив в результате различные результирующие данные.

Передача данных между блоками реализуется посредством связи между выходом первого и входом второго блока. Связь – это класс-наследник интерфейса IPipe (приложение Д). Также следует учитывать очередность выполнения блоков в алгоритме. Пользователь может изменить очередность выполнения блоков, не изменяя при этом связей между ними. Данная возможность доступна путем изменения свойства ExecutionOrder в классе блока.

Когда алгоритм выстроен, пользователь может запустить его и посмотреть значения выходов каждого блока. Таким образом, у пользователя есть возможность локализовать ошибку при формировании алгоритма.

Помимо формирования алгоритма обработки данных пользователь может сконфигурировать параметры отчета на основе обработанных данных. Отчет может содержать данные из одного или более алгоритмов. Пользователь работает с теми данными, которые доступны при его роли в системе, заданной администратором веб-портала. Данные могут быть визуализированы различным образом: таблицы, графики, диаграммы и т.д. Пользователь выбирает вид визуального представления данных и привязывает к нему обработанные данные. Такой подход к проектированию отчета четко отображает требуемую пользователю информацию в удобной для него форме.

По окончании формирования структуры отчета пользователь может сохранить его конфигурацию в базе данных. При внесении изменений в эту конфигурацию пользователь перезапишет данные редактируемой конфигурации отчета. Параметры алгоритма обработки данных также записываются в базу данных.

Рассмотрим функционал системы при реализации второго метода на диаграмме вариантов использования с точки зрения администратора (рис. 8).

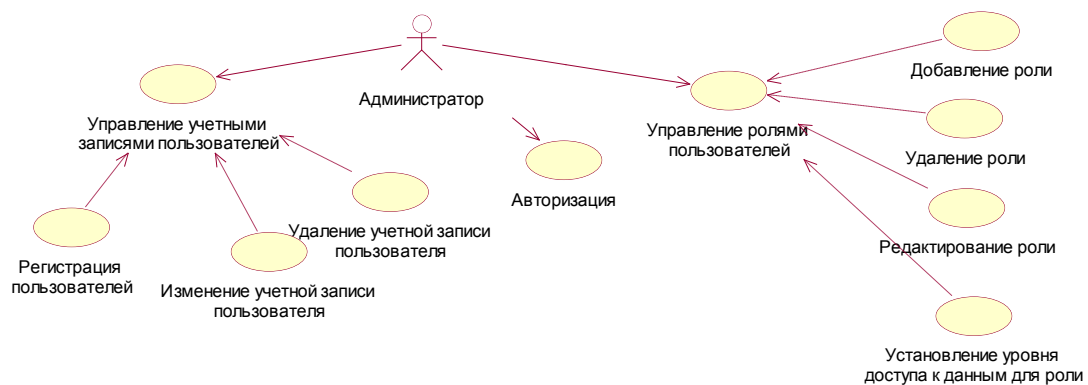


Рисунок 8. Диаграмма вариантов использования системы при реализации метода формирования отчетов №2 с точки зрения администратора.

Главной целью администратора является управление учетными записями пользователей, а именно: добавление, изменение и удаление учетных записей пользователя. После регистрации пользователя администратор веб-портала выдает ему логин и пароль. У пользователя есть возможность изменить выданный пароль.

При настройке ролей пользователей администратор указывает наименование роли и список таблиц, к которым пользователь имеет доступ.

Администратор входит на веб-портал под индивидуальным логином и паролем и только после этого получает доступ к редактированию данных.

Таким образом, администратор не участвует в непосредственном создании алгоритмов обработки данных или в формировании отчетов, но регулирует иерархию доступа пользователей к данным.

Диаграмма последовательности, представленная на рисунке 9, демонстрирует алгоритм формирования отчета. Как было отмечено ранее, конфигурация отчета сохраняется в базу данных конфигураций. При запуске отчета его конфигурация подгружается из базы данных, т.е. на основе этих данных создается модель для анализа данных и модель структуры отчета. Получив эти модели, веб-приложение сначала делает выборку из источников данных по установленным пользователем параметрам, затем обрабатывает эти данные при помощи блоков операций. Уже обработанные данные, готовые для отображения, передаются модели, формирующей отчет, а сама модель передается представлению. Содержащиеся данные в этой модели будут визуализированы в соответствии с установленными пользователем требованиями, которые также передаются представлению в виде конфигурации отчета.

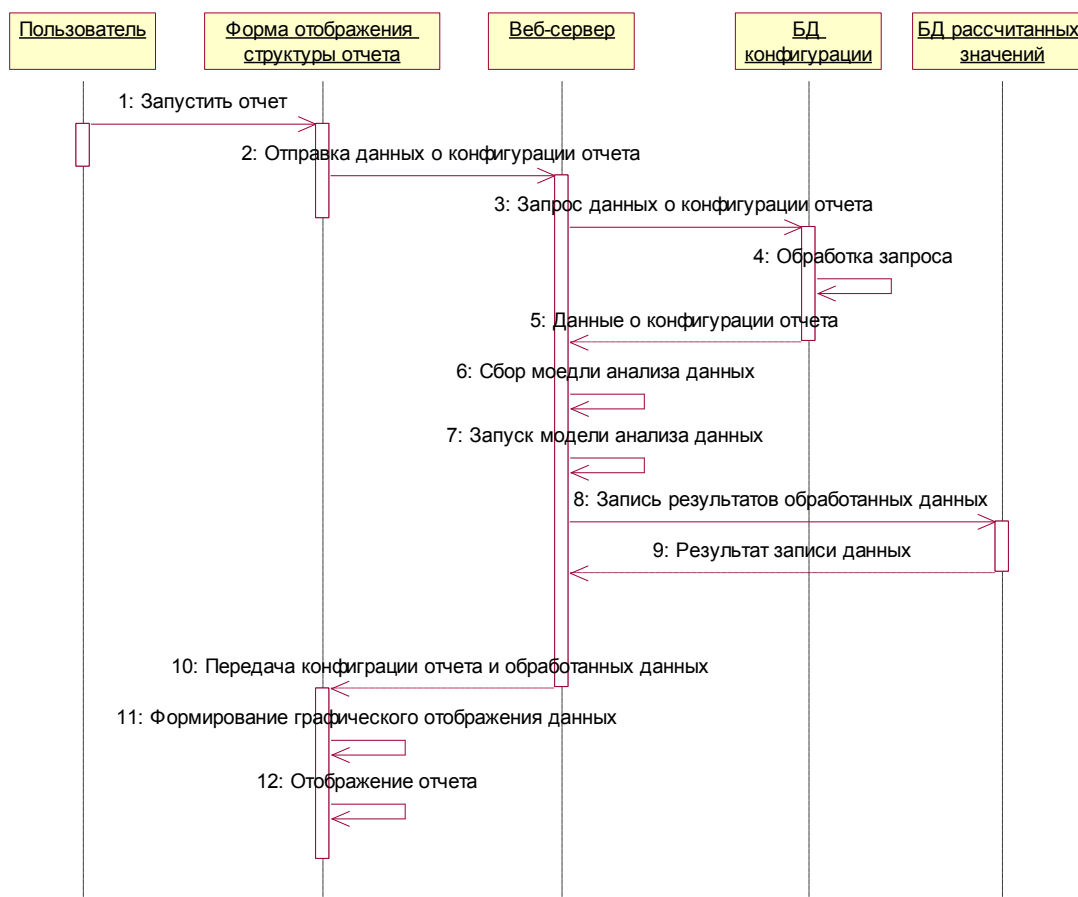


Рисунок 9. Диаграмма последовательности обработки данных и формирования отчета.

Данный алгоритм позволяет избежать повторного запроса данных из источника данных и их обработки, если пользователи захочет просмотреть ранее отработавший запрос. Количество извлекаемых для анализа данных может быть велико, что скажется на длительности формирования отчета.

Выводы по главе

При использовании метода динамического формирования отчетной документации пользователь может самостоятельно конфигурировать алгоритм обработки данных и формировать структуру отчета. Он должен обладать информацией о конфигурации источников данных, о структуре хранящихся в них данных, но не обязательно обладать знаниями в области программирования. При использовании второго варианта системы пользователь должен четко формулировать требования к отчету. Но в этом случае срок разработки или доработки программного кода по формированию отчета может быть неопределенным. По причине наглядности такого алгоритма не составит труда ознакомиться с логикой обработки данных и внести коррективы, удалив, добавив блок или изменив его конфигурацию.

Таким образом, метод формирования отчетов №2 предоставляет пользователю многофункциональную платформу для построения алгоритма обработки данных и формирования структуры отчета. Такой подход также не требует от пользователя знания языка программирования. Также при применении данного метода возможно сократить время на составление алгоритма отчета, поскольку нет необходимости составлять список требований к отчету и передавать его стороннему лицу, что может быть экономически выгодно для компании, а также позволит сократить время на разработку отчета. При такой архитектуре системы стоит также отметить широкий функционал администратора, позволяющий учитывать уровень доступа пользователя системы к данным. Недостатком такого метода является возможная сложность в использовании системы на начальных этапах ее внедрения, поскольку пользователю необходимо уяснить основную концепцию построения алгоритмов. На основании проведенного анализа методов формирования отчетных документов метод динамическо-

го формирования отчетной документации принят для реализации в веб-портале «SmartReports».

Глава 3. Программная реализация методов динамического формирования отчетов

Разработанные методы и алгоритмы реализованы при создании веб-портала «SmartReports» компании «Элеси».

3.1 Цели создания веб-портала SmartReport

Веб-портал «SmartReports» способен повысить эффективность работы предприятия за счет снижения временных затрат на поиск отчетных документов. Таким образом, доступ ко всем отчетным документам будет осуществляться посредством веб-портала. «SmartReports» способствует повышению эффективности управления предприятием за счет своевременного мониторинга и анализа предоставляемой информации о производственных и технологических показателях. Следовательно, увеличивается информированность персонала компании о технологических и производственных процессах компании.

3.2 Описание веб-портала SmartReport

Веб-портал «SmartReports» представляет собой веб-приложение, доступное в корпоративной сети предприятия. Он состоит из двух компонентов:

- серверная часть – обеспечивает выполнение бизнес-логики приложения и предоставление информации клиентам по запросу;
- клиентская часть – предназначена для отображения контента веб-портала, обеспечивает взаимодействие пользователя с веб-приложением.

Для реализации серверной и клиентской части веб-портала использовались разные технологии, поскольку они выполняют различные функции.

В качестве шаблона реализации веб-приложения была выбрана технология ASP.NET MVC4 и язык C#. База данных веб-портала реализована в объектно-реляционной СУБД PostgreSQL версии 9.5. Эта база данных является свободно распространяемой и не требует лицензирования. Для связи веб-приложения с базой данных используется поставщик данных Npgsql с версией не ниже 3.0.

В открытом доступе есть библиотека, организующая алгоритм управления учетными записями пользователей и ролями – «ASP.NET Membership Pro»

vider and Role Provider». Данная библиотека доступна на сайте www.nuget.org, а также ее можно установить в разрабатываемый проект через NuGet Manager. При разработке веб-портала «SmartReports» использовалась данная библиотека. При первичном запуске проекта с добавленной на нее ссылкой в строке подключения создаются таблицы, необходимые для работы с учетными записями пользователя.

Для реализации клиентской части веб-портала использовались свободно распространяемые библиотеки Flotr2, ChartJS, содержащие набор CSS-стилей и JavaScript-кода для графического отображения данных в виде графиков, таблиц и т.д. Сами веб-страницы написаны на языке HTML с применением различных CSS-стилей, в том числе с использованием библиотеки Metro, содержащей набор CSS-стилей JavaScript-файлов по построению интерфейса веб-страницы в стиле Windows 8.

3.3 Программная реализация динамического формирования отчетов

В главе 2 описан широкий функционал пользователя при применении метода динамической обработки данных. Рассмотрим последовательно операции по составлению алгоритма обработки данных.

Алгоритм обработки данных

После успешной авторизации в веб-портале пользователь может перейти к редактированию списка конфигураций (рис. 10). Как уже было упомянуто выше, конфигурация представляет собой логическую структурную единицу, содержащую совокупность отчетов и алгоритмов обработки данных.

На форме редактирования списка конфигураций можно создать новую конфигурацию, переименовать или удалить конфигурацию. После внесения изменений в список конфигураций необходимо сохранить изменения в базе данных.

Для отображения хранящихся данных о конфигурациях, подсистемах, схемах, блоках и т.д. применяется технология EntityFramework.

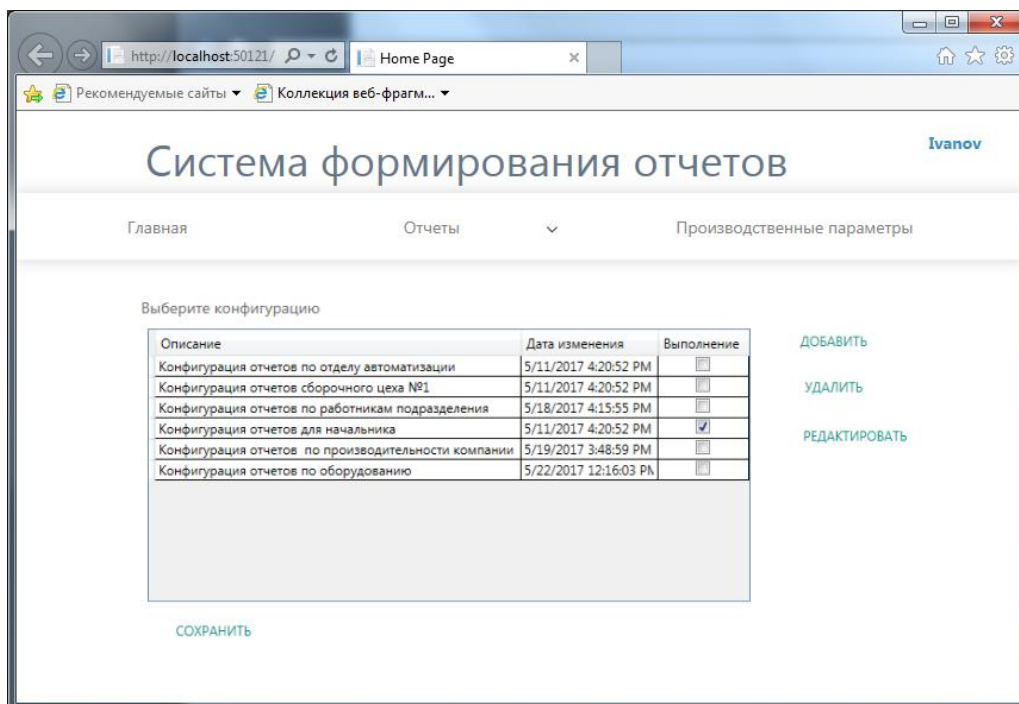


Рисунок 10. Начальная страница работы с конфигурациями.

Далее пользователь может перейти к редактированию выбранной конфигурации. Конфигурация содержит ряд подсистем. Подсистемы формируются в соответствии с производственными единицами, по которым составляется отчет (отдел менеджмента предприятия, сварочный цех №1, система водоснабжения Кировского района и т.д.). В данном случае рассматривается подсистема оборудования предприятия. На рисунке 11 изображена подсистема и созданная в ней схема алгоритма обработки данных.

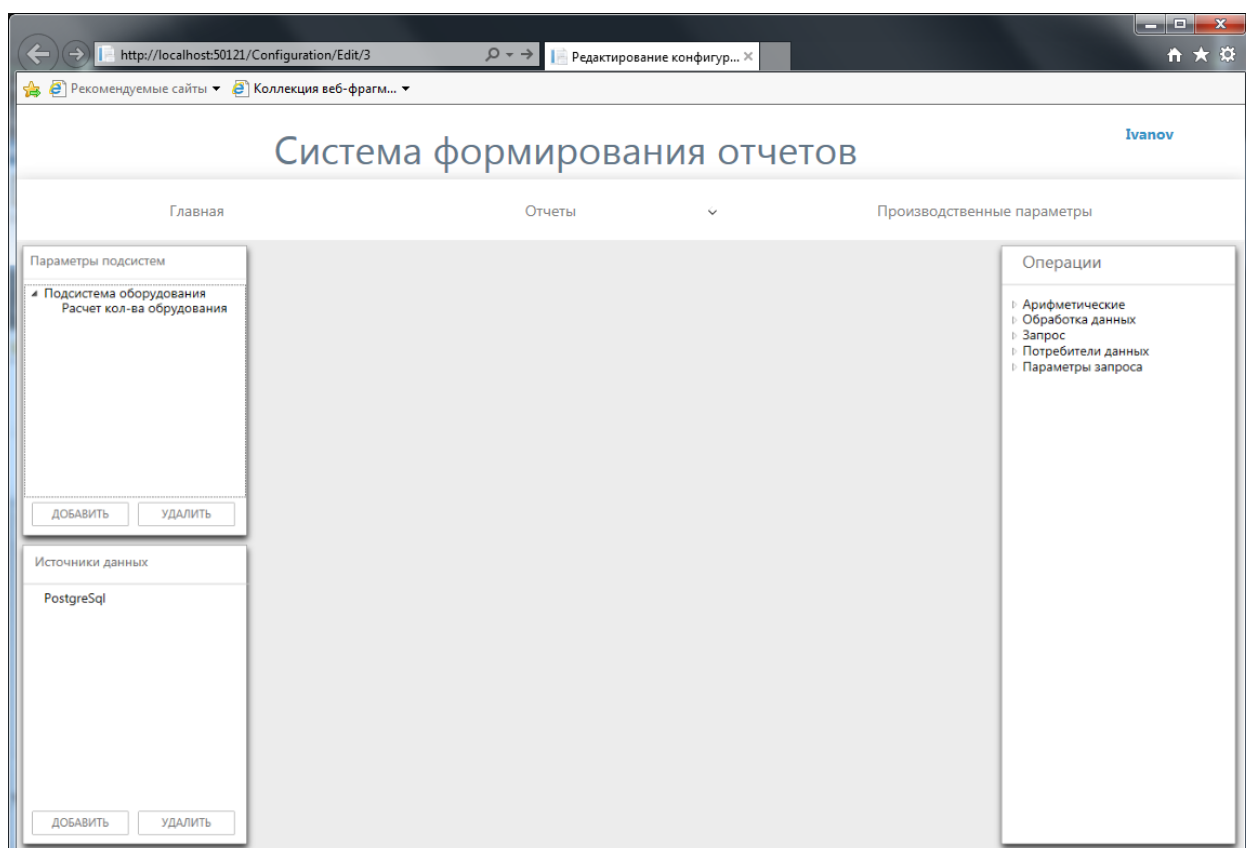


Рисунок 11. Подсистемы в конфигурации.

В рамках одной подсистемы может быть создано несколько схем, т.е. несколько алгоритмов по обработке данных. Алгоритм по обработке данных состоит из последовательности блоков, соединенных между собой связями.

При двойном щелчке на схему в центральной части страницы открывается блок-схема с возможностью создания алгоритма по обработке данных. Схема представляет собой `PartialView` «`SchemePartialView`». Данное частичное представление будет содержать блоки алгоритма по обработке данных. Для включения частичного представления в страницу редактирования требуется указать ссылку на него следующим образом: `@Html.Partial("~/Views/Scheme/SchemePartialView.cshtml")`

При визуализации блоков на веб-странице использовалась библиотека `Metro UI CSS`. Так, для блоков происходил подсчет входов и выходов статическим классом `PinManager`, выполняющим основные операции над входами и выходами. На основании этих данных вычислялась высота элемента `div` со стилем `<grid>`. Таким образом, при открытии схемы с блоками по обработке данных происходило динамическое создание элементов типа `div`.

```

@{
foreach(var item in scheme.GetAll()){
if(item is OperationBase)
{
    var block = item as OperationBase;
    int pinCount = PinManager.GetPinCount(block);
    int blockHeight = (pinCount + 1)*40;
    blockHeight = 200;
    <div class="grid">
        <div class="row">
            <div class="cell" width=@blockWidth height=@blockHeight></div>

            <div class="cell">
                for (var i = 1; i <= PinManager.GetInputCount(item); i++)
                {
                    var pin = PinManager.GetPinByNumber((item as IOperation).Inputs.Values, i);
                    <div class="cycle-button large-button" style="position: fixed; top: @pin.Position.Top
px; left: pin.Position.Left px; width: 30px; height: 30px;"/>
                }

                for (var i = 1; i <= PinManager.GetOutputCount(item); i++)
                {
                    var pin = PinManager.GetPinByNumber((item as IOperation).Inputs.Values, i);
                    <div class="cycle-button large-button" style="position: fixed; top: @pin.Position.Top
px; left: pin.Position.Left px; width: 30px; height: 30px;"/>
                }
            </div>
        </div>
    </div>
}
else if(item is IPipe)
{
    var pipe = item as IPipe;
    var x1 = pipe.Input.Position.Left;
    var y1 = pipe.Input.Position.Top;
    var x2 = pipe.Output.Position.Left;
    var y2 = pipe.Output.Position.Top;
    <line x1=@x1 y1=@y1 x2=@x2 y2=@y2 />
}
}
}

```

Производится обход элементов схемы, которые можно получить, используя метод `scheme.GetAll()`. Далее в зависимости от типа элемента происходит динамическое создание

Далее визуализируются входы и выходы блоков также при помощи элемента `div`, но для их отображения в виде кнопок применяется класс «`cycle-button large-button`». Координаты входов и выходов

При построении древа конфигураций происходит динамическое формирование его ветвей. Для визуализации элемента `TreeView` использовались классы `front-end` библиотеки `Metro UI CSS`.


```

<div class="treeview" data-role="treeview">
  <ul>
    <li class="node">
      @{
        foreach (var system in Config.Systems)
        {
          <span class="leaf">system.Description</span>
          <span class="node-toggle"></span>
          <ul>
            foreach (var scheme in system.Schemes)
            {
              <li><span class="leaf">scheme.Description</span></li>
            }
          </ul>
        }
      }
    </li>
  </ul>
</div>

```

В правой части формы находится список операций. Операции – это блоки, при помощи которых выстраивается алгоритм по обработке данных. Классы блоков описаны в отдельном проекте «Operations». Скомпилированный файл проекта в формате dll помещается в папку Operations в той же папке, что и файлы проекта веб-портала. При запуске веб-портала класс OperationImporter автоматически загружает описанные классы операций. Дерево операций выстраивается аналогично дереву подсистем.

Все блоки являются наследниками интерфейса IBlock (приложение Б) и подразделяются на две основные категории: запросы и операции.

Блоки запросов осуществляют запрос к выбранному в них источнику данных. Таким образом, в системе одновременно существует один объект источника данных, а ссылки на него хранятся у блоков запросов. Это позволяет контролировать соединения, а также избежать утечки памяти, потребляемой веб-приложением.

Чаще всего базе данных предприятия содержит множество данных с контроллеров, датчиков и других средств автоматизации. Эти данные избыточны и неинформативны. Для улучшения их восприятия необходимо произвести некоторую фильтрацию или преобразование. Для этого применяются блоки операций. На вход приходят данные для обработки – на выходе получаем преобразованные данные.

Блоки веб-приложения реализованы с применением паттерна проектирования Command. Интерфейс IBlock, наследуемый от интерфейса ICommand переопределяют метод Execute(), где описана непосредственная логика обработки входных данных. Пример реализации блока сумма представлен в приложении В.

Блок может содержать один или несколько входов и выходов. Входы и выходы могут быть заданы жестко разработчиком (нельзя изменить тип входных/выходных данных) или же доступны для пользовательских настроек (можно изменить тип входа/выхода, группу входа/выхода). Доступные для пользовательских настроек входы/выходы объединяются в группы. Для каждой группы входов имеется массив возможных типов. Например, группа «Числовые входы» в массиве возможных типов будет иметь типы Integer, Float, Decimal и т.п.

Далее пользователю необходимо лишь выстроить логику алгоритма выборки и обработки данных при помощи блоков. Так, например, есть возможность создания разных видов источников данных (рис. 12).

Наименование источника данных	
Наименование источника данных	PostgreSQL
Тип источника данных	PostgreSQL Database
Сервер	postgres
IP адрес	localhost
Порт	5432
Имя пользователя	postgres
Пароль	Qwer123

Рисунок 12. Создание источника данных.

Выбрав вид источника данных (база данных PostgreSQL или FireBird, OleDb хранилище, сервера OPC DA или OPC HDA, Excel файл), нужно заполнить параметры подключения к нему. Эти параметры индивидуальны для каждого источника данных. Для верификации параметров подключения есть возможность проверить соединение к источнику данных.

Далее на схему обработки данных можно добавить запрос и выбрать для него источник данных. Существует несколько видов запросов: SQL, OPC DA и OPC HDA. Таким образом, при выборе источника данных запроса производится их фильтрация по типу. Так, например, для SQL-запроса можно выбрать только источники данных типа PostgreSQL, FireBird или OleDb. Затем можно приступить к конфигурации самого запроса.

Блок запроса может иметь выходы, на которые могут поступать параметры для выборки данных. На данный момент сам запрос пишется на языке SQL, но в дальнейшем планируется визуализировать таблицы баз данных, дав пользователю возможность выбирать необходимые поля таблиц.

The screenshot shows a web browser window with the URL `http://localhost:50121/Block/Edit/12`. The page title is "Система формирования отчетов" and the user name is "Ivanov". The navigation menu includes "Главная", "Отчеты", and "Производственные параметры". The main form has the following sections:

- Комментарий:** A text input field.
- Входы:** A table with columns "Наименование", "Тип", and "Группа". A "ДОБАВИТЬ" button is located below the table.
- Выходы:** A table with columns "Наименование", "Тип", "Условие", and "Группа". The "Наименование" field contains "Out", and the "Тип" dropdown is set to "Таблица". A "ДОБАВИТЬ" button is located below the table.
- Текст запроса:** A text input field containing the SQL query `select * from tools`.

At the bottom right of the form, there are two buttons: "СОХРАНИТЬ" and "ОТМЕНА".

Рисунок 13. Конфигурирование блока SQL-запроса.

Значения входов блока могут быть использованы в запросе в качестве параметров. Синтаксис такого запроса указан на рисунке 14. Таким образом, при построении запроса к базе данных пользователь может использовать результаты вычислений предыдущих блоков в качестве параметров запроса.

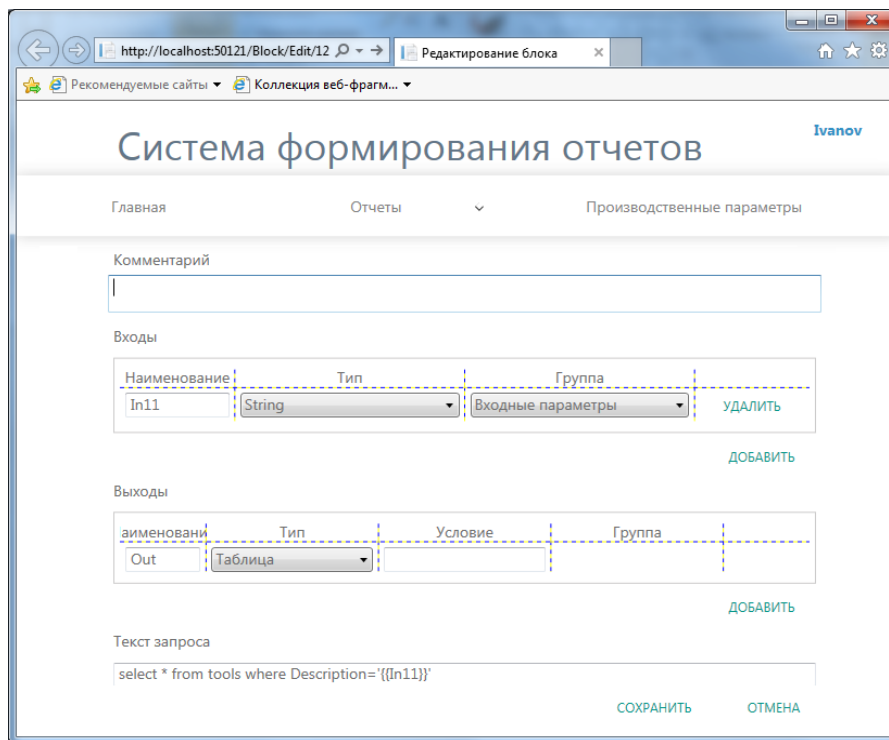


Рисунок 14 Блок sql-запроса с входным параметром.

Здесь `{{In11}}` – это значение входа блока с именем In11. На выход блока поступит результат выполнения запроса в формате DataTable. В дальнейшем эти данные могут быть обработаны блоками операций.

Для передачи значения блоку запроса использовался блок «Константа» с выходом строкового типа (рис. 15).

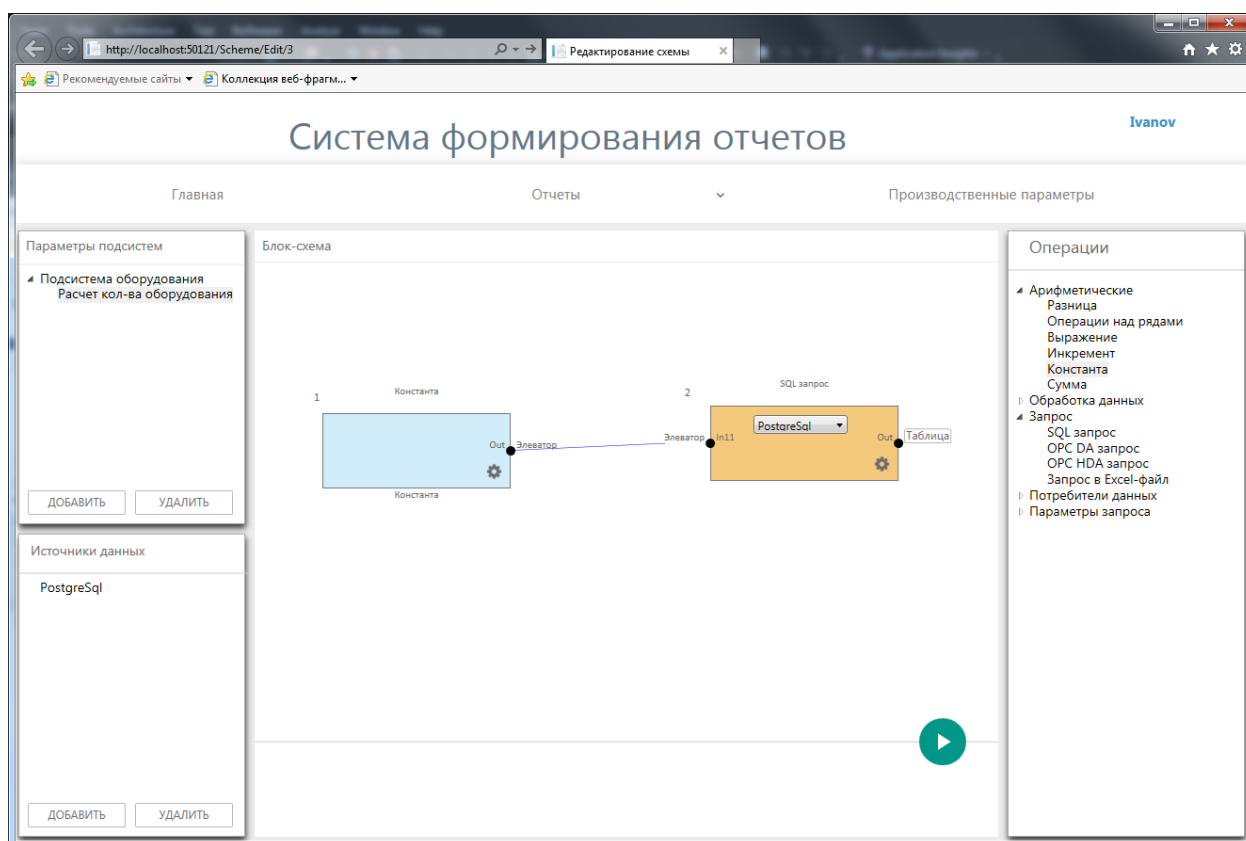


Рисунок 15. Схема с передачей параметра блоку запроса.

По завершении построения алгоритма обработки данных пользователь может запустить его, нажав на кнопку в правом нижнем углу блок-схемы. Нажатие кнопки спровоцирует вызов метода `LaunchScheme()` контроллера «SchemeController».

```
[HttpPost]
public ActionResult LaunchScheme(Scheme scheme)
{
    new ExecuteManager().Execute(OpenedScheme);
}
```

Класс `ExecuteManager` запускает все блоки в соответствии с их очередностью, вызывая у каждого метод `Execute`. В данном методе прописан алгоритм обработки данных, приходящих на вход.

В результате SQL-запроса с параметром из базы данных извлекается одна запись, поле «description» которой принимало значение, переданное блоком «Константа» (рис. 16). Поскольку результатом запроса является таблица, то ее содержимое отображается на отдельной странице.

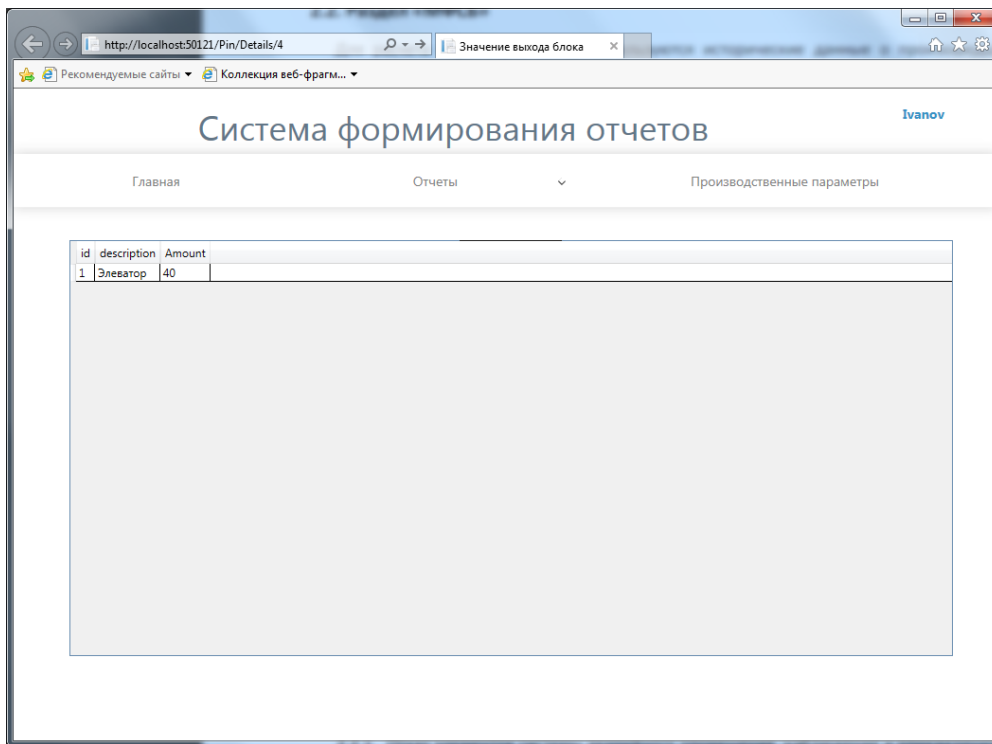


Рисунок 16. Результат выборки блока запроса.

При конфигурировании OPC DA-запроса указывается список тегов на считывание, а также номера свойств тегов (рис. 17).

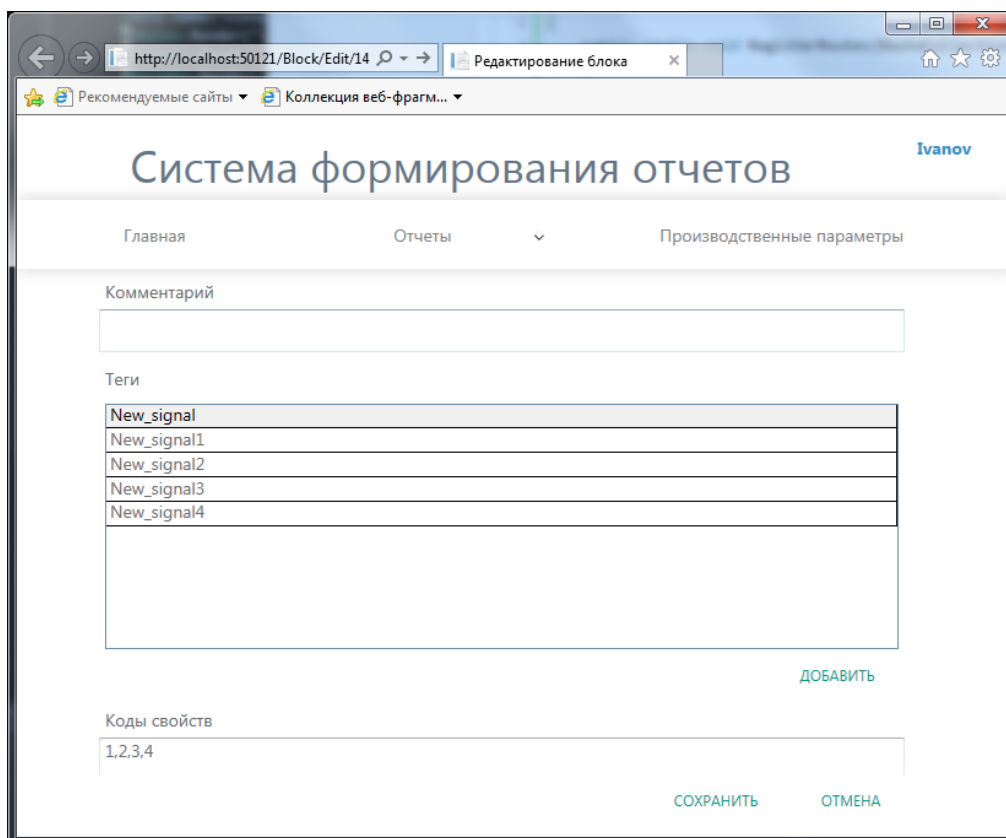


Рисунок 17. Конфигурирование блока OPC DA запроса.

В результате запроса списка сигналов на выход блока попадет словарь считанных тегов, где ключ формируется по шаблону «Наименование тега.[номер свойства]».

Key	Value
New_signal[2]	11
New_signal[3]	goodLocalOverride
New_signal[4]	23.05.2017 15:52:42
New_signal1[2]	4
New_signal1[3]	goodLocalOverride
New_signal1[4]	23.05.2017 15:52:45
New_signal2[2]	12
New_signal2[3]	goodLocalOverride
New_signal2[4]	23.05.2017 15:52:47
New_signal3[2]	6
New_signal3[3]	goodLocalOverride
New_signal3[4]	23.05.2017 15:52:51
New_signal4[2]	7
New_signal4[3]	goodLocalOverride
New_signal4[4]	23.05.2017 15:52:53

Рисунок 18. Результат OPC-блока запроса.

Аналогично происходит выборка из OPC HDA источника данных, но на блок запроса типа OPC HDA поступает период, за который нужно выбрать значения сигналов (дата начала и дата окончания). Учитывая особенности OPC HDA источника данных, номера свойств сигналов указывать необязательно, поскольку хранится только 4 свойства: (1 - Type, 2 - Quality, 3 - Value, 4 - TimeStamp). По этой причине при конфигурировании блока OPC HDA запроса нет возможности указать номера свойств тега.

Выбрав данные, можно произвести некоторые преобразования над ними. На данный момент ведется разработка блоков обработки данных, однако уже есть некоторые основные блоки, такие как обход коллекции, сумма, разница, выражения и т.д. Так, на вход блока операции приходят выбранные блоком запроса данные и проходят некоторую обработку, заданную пользователем.

Рассмотрим блок операции «Сумма» (рис. 19). На вход этого блока поступает таблица типа DataTable. В поле «Условие» выхода блока указано наименование колонки, значения которой должны быть просуммированы. Таким образом, при запуске блока производится обход всех строк таблицы, а значения указанной колонки суммируются и попадают на выход блока «Сумма».

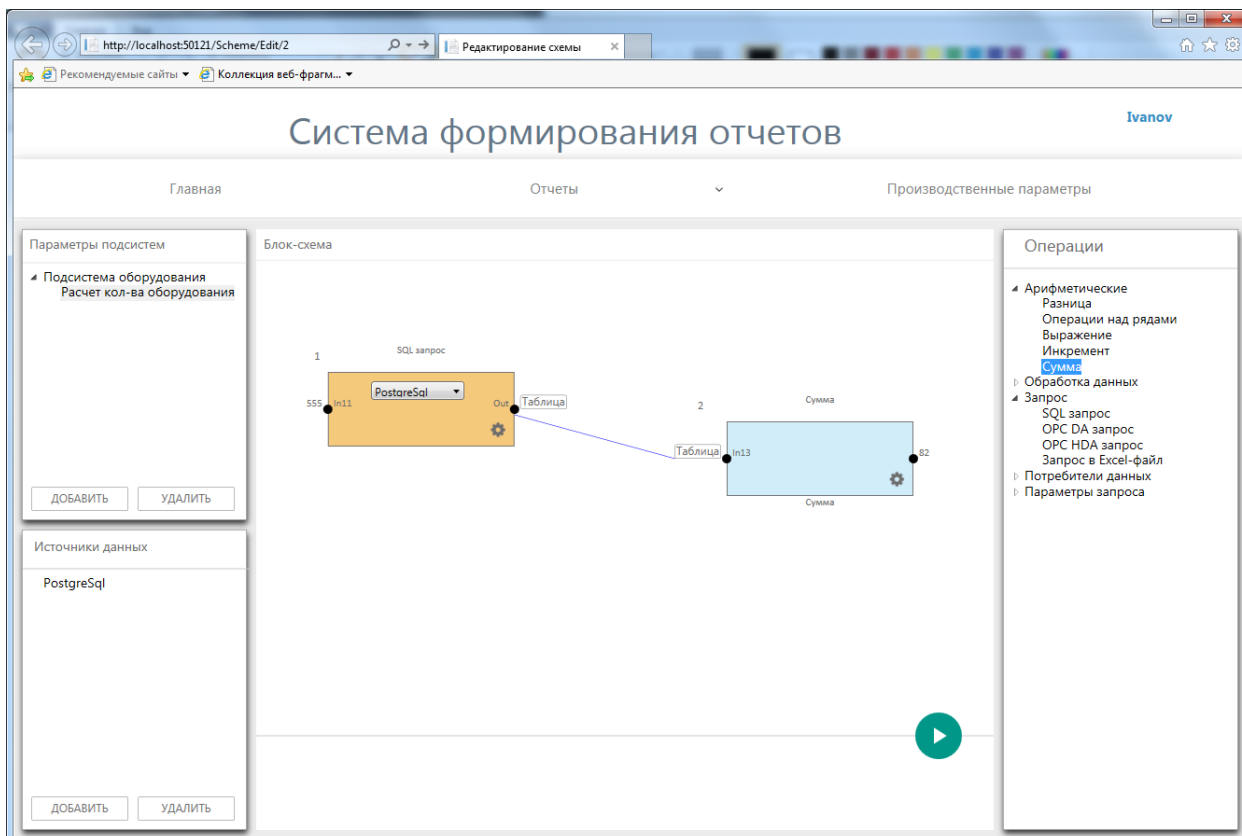


Рисунок 19. Конфигурирование блока операции «Цикл».

Таким образом, выстраивается алгоритм по обработке данных. Результирующие данные попадают в конечную точку (EndPoint). Это элемент записывает полученные данные в базу рассчитанных значений, которые впоследствии будут использованы при построении отчета. На данный момент производится разработка базы данных рассчитанных значений и элемента EndPoint.

Алгоритм построения шаблона отчета

Отчет, принадлежащий определенной подсистеме, использует обработанные данные, записанные элементом EndPoint, для построения на их основе отчетной документации. Все обработанные этой подсистемой данные считываются из базы данных рассчитанных значений и передаются представлению отчета. Далее пользователь может выбрать, какие данные этой подсистемы будут отображаться в отчете.

Система формирования отчетов Петров П. П.

Главная **Отчеты** Производственные параметры

Подсистема "Оборудование"

Таблица "Оборудование" Список тегов "Оборудование" Список тегов "Персонал"

Id	Name	DepartmentName
1	value 1	Цех №1
2	value 2	Цех №2
3	value 3	Цех №1
4	value 4	Цех №5
5	value 5	Цех №1
6	value 6	Цех №2
7	value 7	Цех №4
8	value 8	Цех №1
9	value 9	Цех №3

Тег	Значение
tag1	tagValue1
tag2	tagValue2
tag3	tagValue3
tag4	tagValue4

Тег	Значение
tag1	350
tag2	400
tag3	200
tag4	100
tag5	240

Далее >

Рисунок 20. Выбор данных для построения отчета.

После определения набора данных пользователю предоставляется возможность выбрать визуальное представление данных (рис. 21).

Система формирования отчетов Петров П. П.

Главная **Отчеты** Производственные параметры

Подсистема "Оборудование"

Оборудование Персонал

Radio button Radio button График Гистограмма

Radio button Таблица Круговая диаграмма Таблица

< Назад Далее >

Рисунок 21. Страница выбора графического представления данных.

После выбора пользователем необходимого вида отображения данных происходит автоматическое построение отчета. Контроллер передает модель данных представлению, которое представляет данные в графическом виде (таблица, график и т.д.).

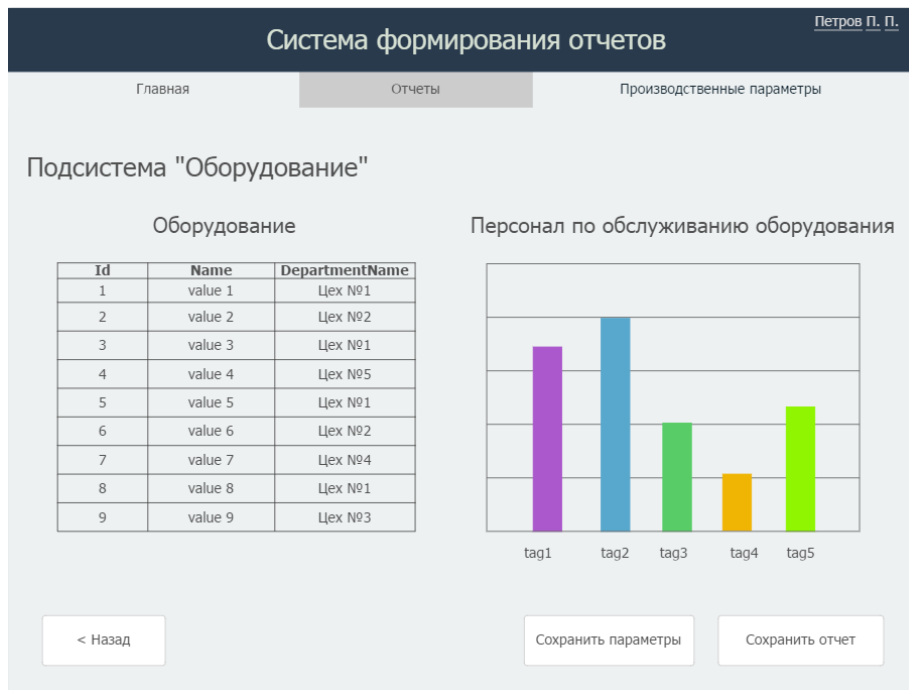


Рисунок 22. Отчет с графическим представлением обработанных данных.

Работу с графиками планируется реализовать при помощи библиотек Flotr2 и ChartJS. Код библиотек находится в открытом доступе и не требует лицензирования. Данные библиотеки являются кроссплатформенными и интегрируются с мобильными устройствами с операционными системами Android и IOS.

По окончании формирования шаблона отчета пользователь сможет сохранить параметры конфигурации шаблона в базу данных. Также веб-портал даст возможность сохранить отчет в формате «.pdf».

3.4 Программная реализация подсистемы администрирования веб-портала

Кроме подсистемы динамического формирования отчетов портала SmartReports был разработан функционал по выполнению администрирования портала.

В части проектирования веб-портала «SmartReports» указано, что пользователями данного веб-приложения являются сотрудники компании и администраторы. Для разграничения доступа к материалам сайта была создана область (Area) «Admin». Созданная область регистрируется посредством класса «AdminAreaRegistration». Область представляет собой структуру папок MVC-шаблона. Так, для области «Admin» есть отдельная папка с контроллерами, представлениями и моделями. Это позволяет не только структурировать код, разбивая его на более мелкие подсистемы, но и управлять маршрутизацией, поскольку созданная область становится также частью URL-адреса к веб-странице проекта. Например, URL-адрес, направляющий на страницу регистрации нового пользователя, выглядит следующим образом: <http://localhost:50121/Admin/Account/Register>. Здесь Admin – название области (Area), Account – название контроллера, Register – название действия (метода контроллера). Таким образом, разграничение функционала пользователей и администратора производится посредством маршрутизации.

Форма авторизации одинакова как для пользователей, так и для администраторов (рис. 23).

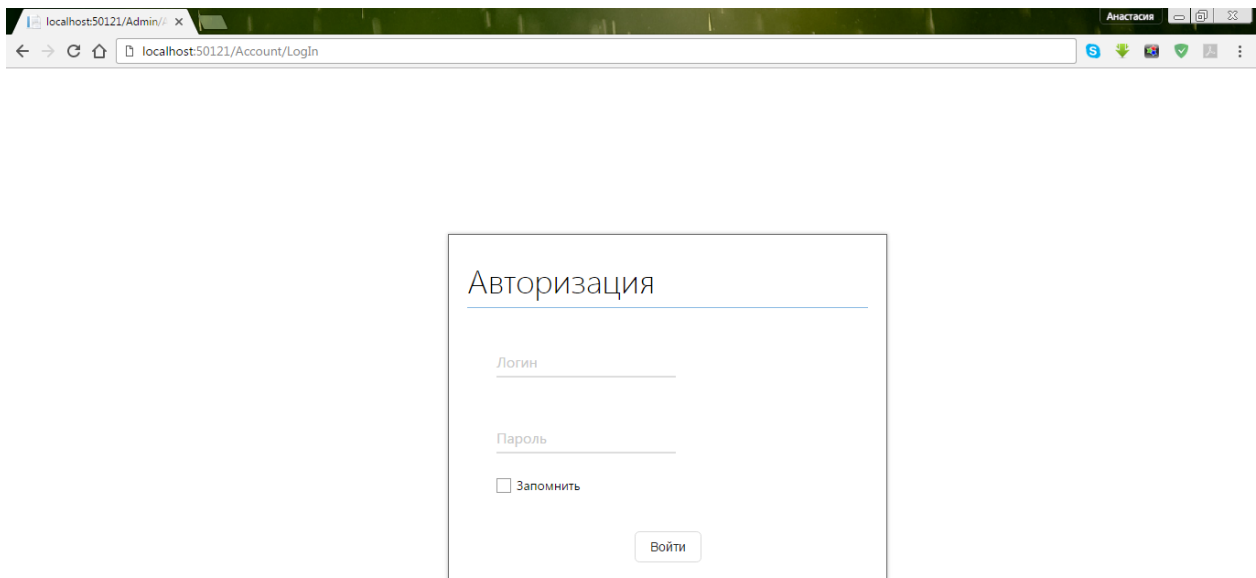


Рисунок 23. Форма авторизации пользователей веб-портала «SmartReports».

Представление данной формы создано с использованием библиотеки Metro, которая содержит элемент управления login-form. Для применения стиля к элементам формы необходимо задать им описанный в библиотеке CSS-класс. Например, элемент ввода логина выглядит следующим образом:

```
<div class="input-control modern text iconic" align="center">  
    @Html.TextBoxFor(m => m.UserName, new { @class = "text" })  
    <span class="label">Логин</span>  
    <span class="informer">Введите логин</span>  
    <span class="placeholder">Логин</span>  
    <span class="icon mif-user"></span>  
</div>
```

Применение готовых стилей для элементов формы сокращает время разработки проекта.

После успешной авторизации, администратор может зарегистрировать новых пользователей в веб-портале. Для администратора веб-портала графический интерфейс создавался без применения каких-либо библиотек стилей, поскольку администратор не является первостепенным пользователем системы.

Далее в этой главе библиотека «ASP.NET Membership Provider and Role Provider» будет рассмотрена более детально. При первичном запуске проекта данная библиотека создает таблицы в базе данных, необходимые для управления персональными страницами и ролями пользователей.

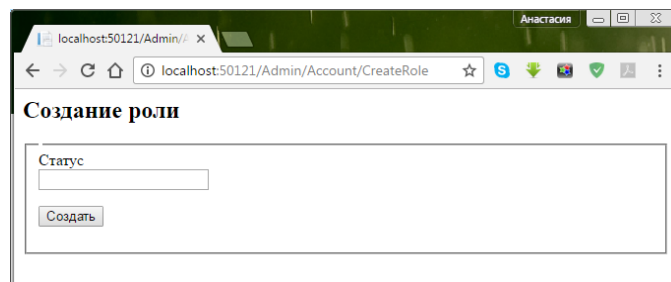


Рисунок 24. Веб-страница создания ролей пользователей.

Таким образом, управление ролями пользователей значительно упрощается, поскольку данная библиотека содержит класс `Roles`. После введения наименования роли на форме (рис. 24) добавление новой записи о роли происходит следующим образом:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult CreateRole(RoleModel roleModel)
{
    if (ModelState.IsValid)
    {
        if (!Roles.RoleExists(roleModel.RoleName))
            Roles.CreateRole(roleModel.RoleName);
        Roles.GetAllRoles();
    }
    return View(roleModel);
}
```

Метод `Roles.RoleExists` проверяет наличие записи о такой же роли в базе данных. Если такая запись отсутствует, то в базу данных записываются данные о роли, после чего происходит обновление списка ролей.

При регистрации нового пользователя (рис. 25) администратор указывает, к какой роли он принадлежит. Также производится верификация пароля: его длина должна быть не менее 6 символов. Данный параметр регулируется на уровне модели следующим образом:

```
[Required(ErrorMessage = "Пароль не может быть пустым")]
[StringLength(100, ErrorMessage = "Введенный пароль должен иметь длину не менее {2}
символов.", MinimumLength = 6)]
[DataType(DataType.Password)]
[Display(Name = "Пароль")]
[Column("password")]
public string Password { get; set; }
```

Свойство Password имеет ряд атрибутов, в том числе StringLength, регулирующее длину хранимой свойством строки, Column – наименование колонки в базе данных, Display – текст элемента на форме, Required – обязательно ли поле для заполнения, DataType – задает имя типа для элемента на форме. Таким образом, элемент ввода пароля – это не просто текстовое поле, а специальный элемент TextBox с функцией скрытия вводимого текста.

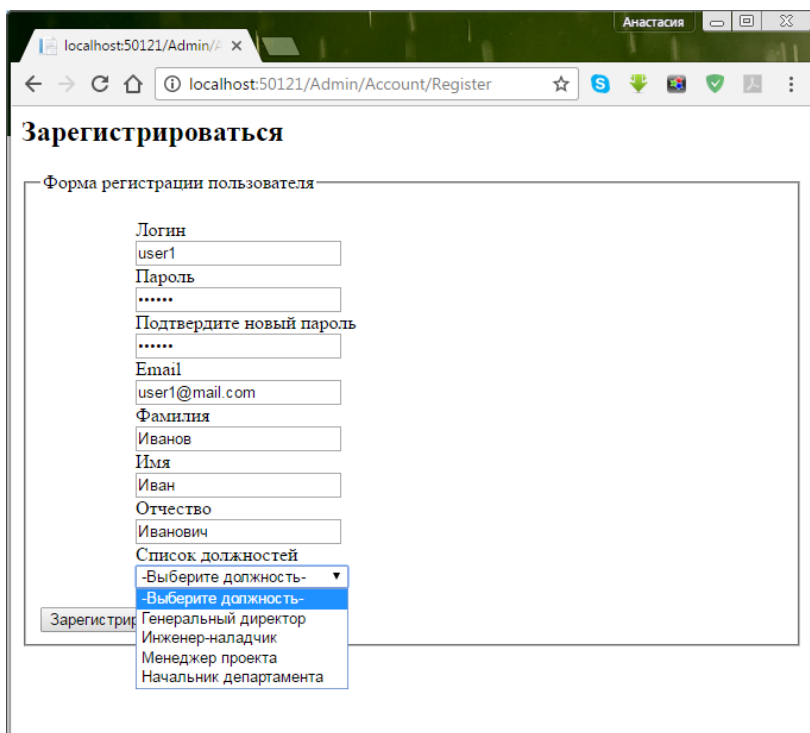


Рисунок 25. Форма регистрации новых пользователей.

После регистрации введенные данные о пользователе сохраняются в базу данных веб-портала в таблице «users» (рис. 26).

user_id	user_name	application_name	last_activity	created	email	salt	password	approved	last_lockout	last_login	last_password	password_question	password_answer
[PK] serial	character varying(25)	character varying(25)	timestamp with time zone	timestamp with time zone	character varying(25)	character varying(25)	bytea	boolean	timestamp with time zone	timestamp with time zone	timestamp with time zone	character varying(25)	character varying(25)
1	Ivanov	pgProvider.Tests	2017-05-14 18:01:09.45+	2017-05-14	user1@mail.	2NUBF6c	<binary data>	TRUE		2017-05-14	2017-05-14		<binary data>

Рисунок 26. Данные о пользователе в базе данных веб-портала.

Данная таблица содержит информацию не только о введенных при регистрации данных пользователя, но и о его деятельности в рамках веб-портала

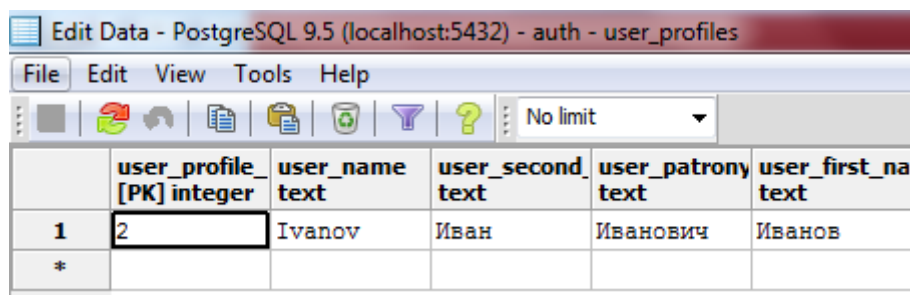
«SmartReports». Так, например, можно узнать дату и время его последней авторизации, дату и время его последней активности на веб-портале, время и дату последнего изменения пароля. Все эти данные записываются автоматически библиотекой «ASP.NET Membership Provider and Role Provider». Запись данных о пользователе производится в одну строку:

```
Membership.CreateUser(userModel.RegisterModel.UserName,  
userModel.RegisterModel.Password, userModel.RegisterModel.UserEmail);
```

Затем для пользователя устанавливается роль:

```
Roles.AddUserToRole(userModel.RegisterModel.UserName, userModel.UserProfile.RoleListValue);
```

Таким образом, использование данной библиотеки максимально облегчает управление учетными записями и ролями пользователей. Для хранения дополнительных сведений о пользователе была создана таблица «user_profiles» (рис. 27).



	user_profile [PK] integer	user_name text	user_second text	user_patronym text	user_first_name text
1	2	Ivanov	Иван	Иванович	Иванов
*					

Рисунок 27. Данные таблицы «user_profiles».

Данная таблица позволяет хранить необходимые данные о пользователе, такие как фамилия, имя, отчество. Связь с таблицей «users» осуществляется посредством логина (user_name), поскольку библиотека по управлению учетными записями и ролями пользователей осуществляет поиск пользователей по логину. Следует уточнить, что логины должны быть уникальны. Запись в эту таблицу производится с использованием технологии EntityFramework.Npgsql v4.0.3.

Класс UsersContext содержит модели, которые могут иметь доступ к базе данных. Для записи данных модели необходимо создать объект этого класса, добавить модель в список объектов моделей этого типа и сохранить произведенные изменения в базе данных.

```

UsersContext db = new UsersContext();
db.UserProfiles.Add(new UserProfile(userModel.RegisterModel.UserName, userMod-
e1.UserProfile.UserFirstName, userMod-
e1.UserProfile.UserSecondName, userMod-
e1.UserProfile.UserPatronym));
db.SaveChanges();

```

EntityFramework позволяет придерживаться объектно-ориентированного подхода даже при работе с базой данных. Это способствует написанию понятного другим разработчикам кода и упрощает процесс дальнейшего изменения (рефакторинга) кода.

3.5 Модель базы данных веб-портала «SmartReports»

Для хранения конфигурации алгоритма обработки данных и шаблона отчета используется база данных PostgreSQL 9.5.

На рисунке 28 представлена модель базы данных, хранящая конфигурацию алгоритма обработки данных и шаблона отчета. Самой крупной структурной единицей является конфигурация, содержащая (Configuration) ряд подсистем (System). Следует отметить, что источники данных едины для целой конфигурации. Источники данных могут быть различных типов, и каждый из них требует индивидуальных параметров для создания строки подключения. По этой причине для каждого типа источника данных существует отдельная таблица с индивидуальными параметрами. Это также облегчает создание объектов источников данных при загрузке конфигурации в веб-приложение.

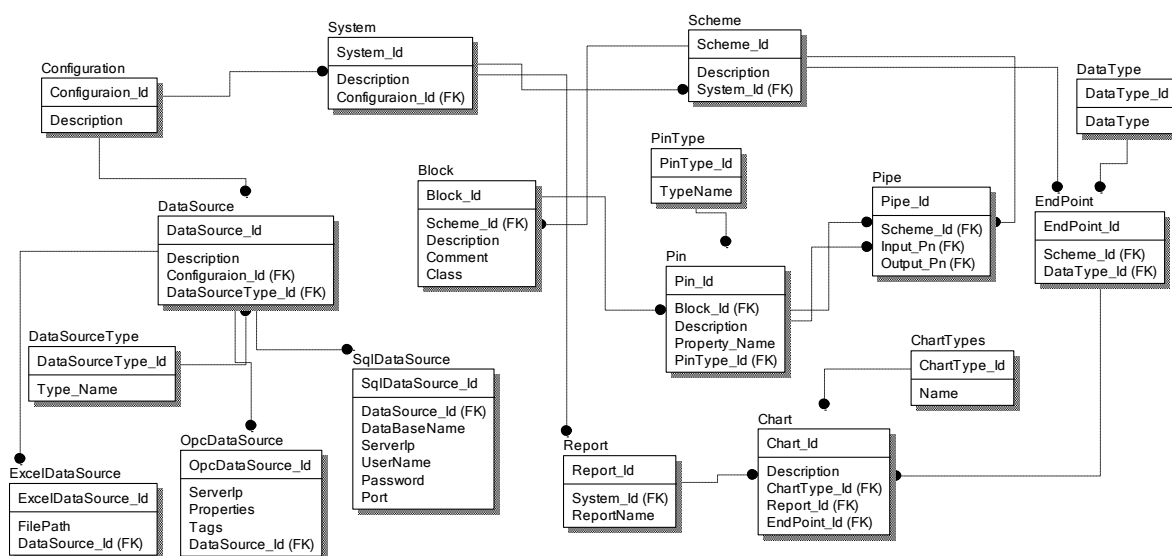


Рисунок 28. Модель базы данных конфигурации алгоритма обработки данных и шаблона отчета.

Подсистема содержит множество схем по обработке данных и множество отчетов. Каждая схема может содержать блоки и связи между ними. Блоки в свою очередь имеют входы и выходы. В общем виде в базе данных они содержатся в таблице Pipe, а их тип – в таблице PipeType. Данная таблица содержит всего два значения – Input и Output. Результирующие обработанные данные попадают на запись. Таблица EndPoint хранит тип результирующих обработанных данных и записывает их в базу данных рассчитанных значений.

Конфигурация отчетов хранится в трех таблицах: Report, Chart, ChartType. Таблица Report содержит описание отчета и ссылки на присутствующие в отчете графики (Chart). В этой таблице хранится вид графического отображения данных (таблица, круговая диаграмма, гистограмма, график и т.д.). Все эти типы описаны в таблице ChartTypes. Также таблица конфигурации графика хранит информацию о типе отображаемых данных (DataTable, Collection<T>, Dictionary<T, T> и т.д.).

Таким образом, централизованное хранение информации о конфигурации алгоритмов обработки данных и шаблонах отчетов позволяет создавать гибкие отчеты на основе уже обработанных данных. Данный подход к формированию отчетов позволяет учесть изменения в алгоритме обработки данных, поскольку есть непосредственная связь между отчетами и конечными обработанными данными. Так, например, если ранее после обработки данных на выходе получали таблицу типа DataTable, а после изменений – Collection<string>, то при построении определенного типа графика веб-портал адаптируется к этому типу данных. При применении метода построения отчетов во втором варианте системы нужно было бы переписывать код, где происходит непосредственное построение графика, т.к. каждый такой файл ориентирован на жестко заданную структуру и при изменении требований как к обработке данных, так и к шаблону отчета, необходимо править программный код.

3.6 Организация работ по разработке портала SmartReports

Веб-приложение «SmartReports» разрабатывается тремя разработчиками компании «ЭлеСи». Вклад данной диссертации в разработку проекта SmartReports состоит в следующем:

- интерфейс взаимодействия с источниками данных (приложение Г);
- функционал конфигурирования блоков запросов и блоков операций
- классы блоков операций (условие, константа)
- классы блоков запросов

На данный момент ведется тестирование подсистемы обработки данных веб-приложения. В будущем планируется усовершенствовать блоки запросов, заменив текст запроса визуальным отображением содержимого источника данных.

Выводы по главе

В результате реализации метода динамического формирования отчетов в корпоративном веб-портале была разработана система «SmartReports». Данный веб-портал состоит из двух подсистем: обработка данных и формирование шаблонов отчетов. На данный момент реализован функционал обработки данных и ведется разработка блоков операций для обработки данных. Корпоративный веб-портал «SmartReports» позволит настроить подключение к источникам данных, производить из них выборку посредством блоков запросов, строить алгоритмы по обработке данных с использованием блоко-операций, а также формировать шаблон отчетов на основе имеющихся обработанных данных.

Глава 4. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

Данный раздел магистерской диссертации производит анализ и оценку технико-экономической эффективности проекта по созданию корпоративного веб-приложения по анализу данных и построению отчетной документации предприятия «SmartReports». На основе результатов произведенных исследований будет сделан вывод об эффективности данного проекта.

4.1 Оценка коммерческого потенциала и перспективности проведения научных исследований и позиции ресурсоэффективности и ресурсосбережения.

4.1.1 Потенциальные потребители результатов исследования.

С целью анализа потребителей результатов научного исследования следует определить целевой рынок и его сегмент, а также критерии сегментирования.

Критериями сегментирования рынка потребителей интернет-услуги являются возраст потребителей, цель использования ресурсов сети Интернет. Приведем карту сегментирования и определим основные сегменты рынка, на которые ориентировано научное исследование по разработке корпоративного веб-портала «SmartReports».

Таблица 1. Карта сегментирования рынка услуг по использованию веб-приложений по анализу данных и формированию отчетной документации.

	Виды деятельности компаний			
		Промышленная компания	Компания сферы услуг	Финансовые компании
Вид рынка	Отечественный рынок	А, Б, В	В	В
	Зарубежный рынок	А, Б	-	-

На данном рынке услуг имеются два конкурента: «WinCC/DataMonitor» компании Siemens и «Ampla» компании Schneider Electric. На карте сегментирования данные компании обозначены, как А и Б соответственно. Фирмой «В» является компания, разрабатывающая данный корпоративный веб-портал «SmartReports».

В приведенной выше карте сегментирования рынка услуг показано, какие ниши на по использованию корпоративных веб-порталов по обработке данных и построению отчетной документации являются приоритетными для исследования на данный момент. На данный момент востребована разработка корпоративных веб-порталов по анализу данных и динамическому формированию отчетной документации для отечественного рынка. Такого плана веб-приложение будет востребовано на любом типе предприятия. Таким образом, наша компания будет иметь возможность занять нишу разработки веб-приложений по анализу данных и формированию отчетной документации на отечественном рынке и будет востребована компаниями с любым видом деятельности.

4.1.2 Анализ конкурентных технических решений.

Проблема конкуренции стоит остро не только в западных странах, но и в России. Особенно важен аспект выведения новых товаров и услуг на рынок, способных удовлетворить потребности клиентов. Данный анализ позволит объективно оценить конкурентов и скорректировать ход научного исследования. Анализ проводится при помощи оценочной карты, предоставляющей анализ нескольких конкурентов, где B_{ϕ} – вес показателя по отношению к фирме, $B_{к1}$, $B_{к2}$ – вес показателя по отношению к конкурентам, K_{ϕ} – конкурентоспособность научной разработки, $K_{к1}$, $K_{к2}$ – конкурентоспособность конкурента. - Конкурентами являются такие системы, как «WinCC/DataMonitor» компании Siemens и «Ampla» компании Schneider Electric.

Таблица 2. Оценочная таблица для сравнения конкурентных технических разработок.

Критерии оценки	Вес критерии	Баллы			Конкурентоспособность		
		B_{ϕ}	$B_{к1}$	$B_{к2}$	K_{ϕ}	$K_{к1}$	$K_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1. Графический интерфейс пользователя	0,1	4	5	4	0,4	0,5	0,4
2. Функциональные возможности	0,1	4	5	4	0,4	0,5	0,4
3. Скорость загрузки веб-страницы	0,05	5	3	5			
4. Защищенность данных	0,1	5	4	4	0,5	0,4	0,4

5. Гибкость системы под нужды пользователя	0,1	5	3	4	0,5	0,3	0,4
6. Степень соответствия современным требованиям	0,05	5	4	4	0,25	0,2	0,2
Экономические критерии оценки эффективности							
1. Уровень проникновения на рынок	0,1	4	5	5	0,4	0,5	0,5
2. Цена	0,1	5	3	2	0,5	0,3	0,2
3. Послепродажное обслуживание	0,1	5	5	4	0,5	0,5	0,4
4. Финансирование научной разработки	0,1	5	3	4	0,5	0,3	0,4
5. Срок выхода на рынок	0,05	5	4	4	0,25	0,2	0,2
6. Наличие сертификации разработки	0,05	5	3	3	0,25	0,15	0,15
Итого	1	Конкурентоспособность			4,45	3,85	3,65

Коэффициент конкурентоспособности фирмы относительно конкурента рассчитывается по следующей формуле: $K_{kc} = \frac{K_{\phi}}{K_{kn}}$.

При расчете коэффициента конкурентоспособности научного исследования относительно «WinCC/DataMonitor» компании Siemens получим: $K_{kc} = \frac{4,45}{3,85} = 1,15$, а относительно и «Ampla» компании Schneider - $K_{kc} = \frac{4,45}{3,65} = 1,22$. Коэффициенты конкурентоспособности научного исследования относительно обоих конкурентов имеют значения больше единицы, следовательно, можно говорить о конкурентоспособности данной работы, что дает возможность заинтересовать данным предложением партнеров и инвесторов.

4.1.3 Технология QuaD

Технология QuaD – это инструмент, созданный с целью измерения и описания характеристик качество новой разработки, ее перспективность на рынке, что позволяет принимать решение о целесообразности вложения денежных средств в научно-исследовательский проект. Целесообразно производить QuaD-анализ научного исследования при помощи оценочной таблицы.

Таблица 3. Оценочная карта для сравнения конкурентных технических (разработок)

Критерии оценки	Вес критерия	Баллы	Максимальный балл	Относительное значение (3/4)	Средневзвешенное значение (5x2)
1	2	3	4	5	

Показатели оценки качества разработки					
1. Графический интерфейс пользователя	0,1	80	100	0,8	0,08
2. Функциональные возможности	0,1	80	100	0,8	0,08
3. Скорость загрузки веб-страницы	0,05	95	100	0,95	0,0475
4. Защищенность данных	0,1	90	100	0,9	0,09
5. Гибкость системы под нужды пользователя	0,1	99	100	0,99	0,099
6. Степень соответствия современным требованиям	0,05	100	100	1	0,05
Показатели оценки коммерческого потенциала разработки					
1. Уровень проникновения на рынок	0,1	70	100	0,7	0,07
2. Цена	0,1	100	100	1	0,1
3. Послепродажное обслуживание	0,1	100	100	1	0,1
4. Финансирование научной разработки	0,1	90	100	0,9	0,09
5. Срок выхода на рынок	0,05	85	100	0,85	0,0425
6. Наличие сертификации разработки	0,05	100	100	1	0,05
Итого	1	P _{ср}			0,899

Оценить качество и перспективность разрабатываемой технологии можно, используя формулу $P_{ср} = \sum B_i * B_i$, где $P_{ср}$ – это средневзвешенное значение показателя качества и перспективности научной разработки, B_i – вес показателя (в долях единицы), B_i – средневзвешенное значение i -го показателя. Получим средневзвешенное значение, используя вышеприведенную формулу: $P_{ср} = 0,899 * 100 = 89,99$. Вычисленное значение больше 80, следовательно, можно сделать вывод о перспективности разработки веб-приложения «Финансы».

4.1.4 SWOT-анализ.

SWOT- анализ позволяет обнаружить угрозы, возможности, сильные и слабые стороны исследования, что является комплексным анализом проекта. Рассмотрим вышеприведенные факторы, используя матрицу SWOT.

Таблица 4. Матрица SWOT.

	Сильные стороны научно-исследовательского	Слабые стороны научно-исследовательского
--	--	---

	<p>проекта: С1. Быстрая загрузка страниц С2. Не нуждается в установке С3. Низкая цена С4. Наличие бюджетного финансирования. С5. Защищенность данных С6. Широкая целевая аудитория С7. Поддержка и сопровождение продукта С8. Высокое качество разрабатываемого продукта</p>	<p>проекта: Сл1. Малая популярность приложения Сл2. Мало дополнительных опций Сл3. Экономический спад Сл4. Смена спроса Сл5. Продукты-заменители</p>
<p>Возможности: В1. Расширение функционала приложения В2. Увеличение рекламы продукта В3. Появление дополнительного спроса на новый продукт В4. Снижение тарифного плана используемого домена В5. Повышение стоимости конкурентных разработок</p>	<p>С4В1 За счет бюджетного финансирования есть возможность увеличить функционал веб-приложения, что привлечет клиентов. С1С2С3С5С7С8В3 Быстрота работы приложения, отсутствие необходимости в установке, низкая цена, надежность вводимых данных, поддержка и сопровождение продукта привлекает клиентов. С6В2 При транслировании рекламных роликов возможно увеличить количество пользователей. С6В4 При снижении платы за домен возможно снизить цену для покупателей, что привлечет клиентов. С3С6В5 Если конкурентные компании повысят цену на продукт, клиентов привлечет низкая цена на наш продукт.</p>	<p>Сл1В1 Чтобы увеличить популярность приложения следует расширить функционал веб-приложения и начать рекламную акцию. Сл4В2 Увеличить спрос на веб-приложение можно путем продвижения веб-приложений. Сл4В4 Снижение тарифного плана используемого домена возможно при неблагоприятной экономической ситуации. Сл3Сл5В5 Вследствие повышения стоимости аналогичных веб-приложений конкурентов возможно по причине экономического спада, падения спроса или возникновения более сильного конкурента.</p>
<p>Угрозы: У1. Отсутствие спроса на разрабатываемый продукт У2. Появление более сильного конкурента У3. Уменьшение целевой аудитории У4. Изменение законодательства РФ У5. Несвоевременное</p>	<p>У1С1С3С5 Избежать угрозы отсутствия или падения спроса, разработав веб-приложение, позволяющее быстро обрабатывать запросы пользователя, и безопасно хранить личные данные. Помимо этого стимуляцией спроса является снижение</p>	<p>У1Сл3 При падении или исчезновении спроса на продукт возможно приостановление работы фирмы. У2Сл2 При появлении конкурента следует расширить опции веб-приложения. У3Сл1Сл5 Целевая</p>

финансовое обеспечение научного исследования со стороны государства	стоимости продукта. У2С3С6При появлении серьезного конкурента возможно снижение цены, расширение целевой аудитории. У3С3С7При уменьшении целевой аудитории следует снизить цену на продукт, выпустить обновление продукта.	аудитория может уменьшится вследствие нахождения веб-приложений со схожим функционалом, малой популярностью веб-приложения, отсутствия возможности подключения к Интернет-ресурсу. У4Сл1При изменении законодательных актов РФ клиенты откажутся от пользования данной продукцией. У5Сл2Сл3 Несвоевременные выплаты со стороны государства могут привести к отставанию в от плана разработки новых опций приложения, а также экономическому спаду фирмы.
---	--	---

Выявление соответствия сильных и слабых сторон научно-исследовательского проекта с целью определения необходимости проведения стратегических изменений следует производить с использованием интерактивной матрицы проекта.

Таблица 5. Интерактивная матрица проекта.

	Сильные стороны проекта									Слабые стороны проекта				
		С1	С2	С3	С4	С5	С6	С7	С8	Сл1	Сл2	Сл3	Сл4	Сл5
Возможност и проекта	В1	-	-	-	+	-	-	-	0	+	-	-	-	-
	В2	-	-	0	-	-	+	-	-	-	-	-	+	-
	В3	+	+	+	-	+	-	+	+	-	-	-	-	-
	В4	0	-	-	-	-	+	-	-	-	-	-	+	-
	В5	-	-	+	-	-	+	-	-	-	-	+	-	+
	У1	+	0	+	-	+	-	-	-	-	-	+	-	-
	У2	-	-	+	-	-	+	-	-	-	+	-	-	-
	У3	-	-	+	-	-	-	+	-	+	-	-	-	+
	У4	-	-	-	-	-	-	-	-	+	-	-	-	-
	У5	-	-	-	-	-	-	-	-	-	+	+	-	-

В результате анализа сильных и слабых сторон разрабатываемого приложения, его возможностей и угроз в соответствии с интерактивной матрицей проекта возможны следующие направления реализации проекта: С4В1,

C1C2C3C5C7B3, C6B2, C6B4, C3C6B5, Сл1B1, Сл4B2, Сл4B4, Сл3Сл5B5, У1С1С3С5, У2С3С6, У3С3С7, У1Сл3, У2Сл2, У3Сл1Сл5, У4Сл1, У5Сл2Сл3.

4.2 Планирование научно-исследовательских работ.

4.2.1 Структура работ в рамках научного исследования.

Ниже приведен комплекс работ по выполнению магистерской диссертации. К исполнителям проекта относятся: Сенина А. А., Тузовский А. Ф. (научный руководитель).

Таблица № 7. Перечень этапов, работ и распределение исполнителей.

Основные этапы	№ раб	Содержание работ	Должность исполнителя
Разработка технического задания	1	Составление и утверждение технического задания	Руководитель проекта
Выбор направления исследования	2	Подбор и изучение материалов по теме	Студент
	3	Изучение существующих объектов проектирования	Студент
	4	Календарное планирование работ	Руководитель, Студент
Теоретическое и экспериментальное исследование	5	Проведение теоретических расчетов и обоснований	Студент
	6	Построение макетов (моделей) и проведение тестирования	Студент
	7	Сопоставление результатов тестирования с теоретическими исследованиями	Студент
Обобщение и оценка результатов	8	Оценка эффективности полученных результатов	Руководитель, Студент
	9	Определение целесообразности проведения ОКР	Руководитель, Студент
Разработка технической документации и проектирование	10	Проектирование корпоративного веб-портала «SmartReports» с использованием технологии ASP.NET MVC	Студент
	11	Составление схемы базы данных	Студент
	12	Составление схемы информационных потоков	Студент
	13	Разработка UML-диаграмм	Студент
	14	Разработка корпоративного веб-портала «SmartReports»	Студент

Оформление отчета	18	Составление пояснительной записки	Студент
-------------------	----	-----------------------------------	---------

4.2.2. Разработка графика проведения научного исследования.

Для удобства построения графика, длительность каждого из этапов работ необходимо перевести из рабочих дней в календарные дни. Для этого необходимо рассчитать коэффициент календарности по следующей формуле:

$$k_{\text{кал}} = \frac{T_{\text{кал}}}{T_{\text{кал}} - T_{\text{вых}} - T_{\text{пр}}} = \frac{365}{365 - 118} = 1,48$$

Таблица №8. Временные показатели проведения работ.

Название работы	Трудоемкость работы									Длительность работ в рабочих днях T_{ri}			Длительность работ в календарных днях, T_{ki}		
	t_{min} , чел-дни			t_{max} , чел-дни			$t_{ож}$, чел-дни			Исп1	Исп2	Исп3	Исп1	Исп2	Исп3
	Исп1	Исп2	Исп3	Исп1	Исп2	Исп3	Исп1	Исп2	Исп3						
Составление и утверждение ТЗ (Р)	5	7	5	6	9	7	5,4	7,8	5,8	5,4	7,8	5,8	7,99	11,54	8,58
Подбор и изучение материала по темам (С)	3	5	6	7	7	7	4,6	5,8	6,4	4,6	5,8	6,4	6,81	8,58	9,47
Изучение существующих решений (С)	3	2	5	5	4	10	3,8	2,8	7	3,8	2,8	7	5,62	4,14	10,36
Календарное планирование (Р,С)	1	1	1	2	3	5	1,4	1,8	2,6	0,7	0,9	1,3	1,04	1,33	1,92
Проведение теоретических подсчетов и обоснований (С)	5	4	3	6	5	5	5,4	4,4	3,8	5,4	4,4	3,8	7,99	6,51	5,62
Построение моделей и проведение тестирования (С)	2	2	1	3	4	4	2,4	2,8	2,2	2,4	2,8	2,2	3,55	4,14	3,26
Сопоставление результатов с теоретическими	1	2	1	4	5	3	2,2	3,2	1,8	2,2	3,2	1,8	3,26	4,74	2,66

На основе таблицы 8 построим календарный план-график. График строится для максимального по длительности исполнения работ в рамках научно-исследовательского проекта. В данном случае максимальная длительность реализации проекта наблюдается исполнитель №3 и составляет 109,8 календарных дней. В таблице 9 приведен календарный план-график с разбивкой по месяцам и декадам (10 дней) за период времени дипломирования.

Таблица №9. Календарный график проведения НИОКР по теме.

№ работ	Вид работ	Исполнители	Продолжительность выполнения работ												
			Февраль	Март			Апрель			Май			Июнь		
			3	1	2	3	1	2	3	1	2	3	1		
1	Составление и утверждение ТЗ (Р)	Руководитель проекта													
2	Подбор и изучение материала по темам (С)	Студент													
3	Изучение существующих решений (С)	Студент													
4	Календарное планирование (Р,С)	Руководитель													
		Студент													
5	Проведение теоретических подсчетов и обоснований (С)	Студент													
6	Построение моделей и проведение тестирования (С)	Студент													
7	Сопоставление результатов с теоретическими исследованиями (С)	Студент													
8	Оценка эффективности полученных результатов (Р,С)	Руководитель													
		Студент													
9	Определение целесообразности проведения ОКР (Р, С)	Руководитель													
		Студент													
10	Разработка «SmartReports» (С)	Студент													

11	Составление схемы базы данных (С)	Студент																		
12	Составление схемы информационных потоков (С)	Студент																		
13	Разработка UML диаграмм (С)	Студент																		
14	Проектирование «SmartReports» (С)	Студент																		
15	Составление пояснительной записки (С)	Студент																		

4.2.3 Бюджет научно-технического исследования (НТИ).

4.2.3.1 Расчет материальных затрат НТИ.

Данный раздел включает стоимость оборудования и программного обеспечения, необходимого для разработки корпоративного веб-портала «SmartReports» с учетом транспортных затрат.

Таблица №10. Материальные затраты НТИ.

Наименование	Единица измерения	Количество			Цена за ед., руб			Затраты на материалы, руб.		
		Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3
Системный блок DNS Extreme 0803807	шт.	1	1	1	2800	28600	28000	33600	34320	33600
Монитор Dexp M180 черный	шт.	1	1	1	4990	5000	4990	5988	6000	5988
Клавиатура+мышь Defenfer Oxford C-975 Nano	шт.	1	0	1	1790	1800	0	2148	0	0
ОС Microsoft Windows 7 Professional бита	шт.	1	1	1	7690	7890	7690	9228	9468	9228
ПО Microsoft Office 365 Home and Business	шт.	1	1	1	8990	8690	8990	10788	10428	10788
Клавиатура Sven Comfort 3050	шт.	0	1	0	570	0	0	0	0	0

Мышь проводная DNS Prestige G-15I	шт.	0	1	0	490	0	0	0	0	0	
Итого:									61752	60216	59604

4.2.3.2 Расчет затрат на специальное оборудование для научных работ.

В данной статье расхода включаются затраты на приобретение специализированного программного обеспечения для разработки корпоративного веб-портала «SmartReports». В таблице №11 приведен расчет бюджета затрат на приобретение программного обеспечения для проведения научных работ:

Таблица №11. Расчет бюджета затрат на приобретение специфичного ПО.

№ п/ п	Наименование оборудования			Кол-во единиц оборудования			Цена единицы оборудования, руб.			Общая стоимость оборудования, руб.		
	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3
1	Visual Studio Professional 2013	Visual Studio Professional 2010	Visual Studio Professional 2012	1	1	1	27345	22450	25890	31446,75	25817,5	29773,5

4.2.3.3 Основная заработная плата исполнителей темы.

Основная заработная плата ($Z_{\text{осн}}$) руководителя (лаборанта, инженера) от предприятия (при наличии руководителя от предприятия) рассчитывается по следующей формуле:

$$Z_{\text{осн}} = Z_{\text{дн}} \cdot T_p,$$

где $Z_{\text{осн}}$ – основная заработная плата одного работника;

T_p – продолжительность работ, выполняемых научно-техническим работником, раб. дн. (табл. 8);

$Z_{\text{дн}}$ – среднедневная заработная плата работника, руб.

Среднедневная заработная плата рассчитывается по формуле:

$$Z_{\text{дн}} = \frac{Z_{\text{м}} \cdot M}{F_{\text{д}}},$$

где $Z_{\text{м}}$ – месячный должностной оклад работника, руб.;

M – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня $M = 11,08$ месяца, 5-дневная неделя;

при отпуске в 48 раб. дней $M = 10,4$ месяца, 6-дневная неделя;

$F_{\text{д}}$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн.

Расчет основной заработной платы приведен в таблице №12.

Таблица №12. Основная заработная плата.

Исполнитель	Оклад, руб.	Районный коэффициент	Среднедневная заработная плата, руб.	Трудоемкость, чел-дн.			Осн. Зароботная плата, руб.		
				Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3
Руководитель	33162,9	1,3	1959,63	6,45	5,05	5,1	12639,61	9896,13	9993,6
Студент	6976,22		412,2	56,25	61,85	69,1	23188	25494,57	28483,02

4.2.3.4 Дополнительная заработная плата исполнителей темы.

Затраты по дополнительной заработной плате исполнителей темы учитывают величину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.).

Расчет дополнительной заработной платы ведется по следующей формуле:

$$Z_{\text{донP(Исп1)}} = k_{\text{дон}} \cdot Z_{\text{осн}} = 0,15 \cdot 12639,61 = 11895,94$$

$$Z_{\text{донC(Исп1)}} = k_{\text{дон}} \cdot Z_{\text{осн}} = 0,15 \cdot 23188 = 3478,2$$

$$Z_{\text{донP(Исп2)}} = k_{\text{дон}} \cdot Z_{\text{осн}} = 0,15 \cdot 9896,13 = 1484,42$$

$$Z_{\text{донC(Исп2)}} = k_{\text{дон}} \cdot Z_{\text{осн}} = 0,15 \cdot 25494,57 = 3824,18$$

$$Z_{\text{допP(Исп3)}} = k_{\text{доп}} \cdot Z_{\text{осн}} = 0,15 \cdot 9993,6 = 1499,04$$

$$Z_{\text{допC(Исп3)}} = k_{\text{доп}} \cdot Z_{\text{осн}} = 0,15 \cdot 28483,02 = 4272,45$$

4.2.3.5 Отчисления во внебюджетные фонды (страховые отчисления).

Отчисления во внебюджетные фонды представлены в таблице №13.

Таблица №13. Отчисления во внебюджетные фонды.

Исполнитель	Основная заработная плата			Дополнительная заработная плата		
	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3
Руководитель проекта	12639,61	9896,13	9993,6	11895,94	1484,42	1499,04
Инженер	23188	25494,57	28483,02	3478,2	3824,18	4272,45
Коэффициент отчисления во внебюджетные фонды, %	30%					
Исп. 1	15360,52					
Исп. 2	12209,81					
Исп. 3	13274,44					

4.2.3.5 Накладные расходы.

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергия, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по следующей формуле:

$$\text{Исп. 1: } Z_{\text{накл}} = (61752 + 31446,75 + 35827,61 + 15374,14 + 15369,52/5) \cdot 0,5 = 15977$$

$$\text{Исп. 2: } Z_{\text{накл}} = (60216 + 25817,5 + 35390,7 + 15374,14 + 12209,81/5) \cdot 0,5 = 14900,81$$

$$\text{Исп. 3: } Z_{\text{накл}} = (59604 + 29773,5 + 38476,62 + 5771,49 + 13274,44/5) \cdot 0,5 = 14690$$

Где 0,5 - коэффициент, учитывающий накладные расходы.

4.2.3.6 Формирование бюджета затрат научно-исследовательского проекта.

Определение бюджета затрат на научно-исследовательский проект приведен в таблице №14.

Таблица №14. Расчет бюджета затрат НИИ.

Наименование статьи	Сумма, руб.		
	Исп. 1	Исп. 2	Исп. 3
1. Материальные затраты	61752	60216	59604
2. Затраты на специальное оборудование	31446,75	25817,5	29773,5
3. Затраты по основной заработной плате исполнителей темы	35827,61	35390,7	38476,62
4. Затраты по дополнительной заработной плате исполнителей темы	15374,14	5308	5771,49
5. Отчисления во внебюджетные фонды	15360,52	12209,81	13274,44
6. Накладные расходы	4410,86	4301,14	4257,43
7. Бюджет затрат НИИ	164171,9	143243,2	151157,5

В соответствии с произведенными расчетами с 2.3.1 - 2.3.6 можно сделать вывод об эффективности исполнения под номером 2 в связи с минимальным бюджетом.

4.3 Определение ресурсной, финансовой, бюджетной, социальной и экономической эффективности исследования.

Интегральный финансовый показатель разработки определяется по формуле

$I_{финр}^{исп.i} = \frac{\Phi_{pi}}{\Phi_{max}}$, где $I_{финр}^{исп.i}$ – интегральный финансовый показатель разработки;

ботки;

Φ_{pi} – стоимость i -го варианта исполнения;

Φ_{max} – максимальная стоимость исполнения научно-исследовательского проекта (в т.ч. аналоги).

$$I_{финр}^{исп1} = \frac{164171,9}{164171,9} = 1$$

$$I_{финр}^{исп2} = \frac{143243,2}{164171,9} = 0,87$$

$$I_{финр}^{исп3} = \frac{151157,5}{164171,9} = 0,92$$

Интегральный показатель ресурсоэффективности вариантов исполнения объекта исследования можно вычислить следующим образом:

$$I_{pi} = \sum a_i \cdot b_i, \quad (16)$$

где I_{pi} – интегральный показатель ресурсоэффективности для i -го варианта исполнения разработки;

a_i – весовой коэффициент i -го варианта исполнения разработки;

b_i^a, b_i^p – бальная оценка i -го варианта исполнения разработки, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Расчет показателя представлен в таблице 15.

Таблица №15. Сравнительная оценка характеристик вариантов исполнения проекта.

Объект исследования / Критерии	Весовой коэффициент параметра	Исп.1	Исп.2	Исп.3
1. Способствует росту производительности труда пользователя	0,2	4	5	3
2. Удобство в эксплуатации (соответствует требованиям потребителей)	0,15	4	3	4
3. Помехоустойчивость	0,1	5	5	4
4. Энергосбережение	0,2	3	4	4
5. Надежность	0,25	3	5	4
6. Материалоемкость	0,1	4	5	4
ИТОГО	1			

$$I_{p-исп1} = 4*0,2 + 4*0,15 + 5*0,1 + 3*0,2 + 3*0,25 + 4*0,1 = 3,65$$

$$I_{p-исп2} = 3*0,2 + 2*0,15 + 3*0,1 + 3*0,2 + 4*0,25 + 4*0,1 = 3,2$$

$$I_{p-исп3} = 4*0,2 + 3*0,15 + 3*0,1 + 3*0,2 + 4*0,25 + 4*0,1 = 3,65$$

Интегральный показатель эффективности вариантов исполнения опре-

деляется по формуле: $I_{исп.1} = \frac{I_{р-исп1}}{I_{финр}}$, $I_{исп.2} = \frac{I_{р-исп2}}{I_{финр}}$ и т.д.

$$I_{исп1} = \frac{3,65}{1} = 3,68$$

$$I_{исп2} = \frac{3,2}{0,87} = 3,67$$

$$I_{исп3} = \frac{3,65}{0,92} = 3,97$$

Сравнение интегрального показателя эффективности вариантов исполнения разработки позволит определить сравнительную эффективность проекта (см.табл.15) и выбрать наиболее целесообразный вариант из предложенных. Сравнительная эффективность проекта ($\mathcal{E}_{ср}$):

$$\mathcal{E}_{ср} = \frac{I_{исп.1}}{I_{исп.2}}$$

Таблица 16. Сравнительная эффективность разработки

№ п/п	Показатели	Исп. 1	Исп. 2	Исп. 3
1	Интегральный финансовый показатель разработки	1	0,87	0,92
2	Интегральный показатель ресурсоэффективности разработки	3,65	3,2	3,65
3	Интегральный показатель эффективности	3,68	3,67	3,97
4	Сравнительная эффективность вариантов исполнения	0,93	0,92	1

В ходе выполненных вычислений и сравнительных операций было выявлено наиболее эффективное исполнение проекта. Данным вариантом оказалось исполнение под номером 3.

Вывод

В результате выполнения работы была произведена оценка технико-экономической эффективности проекта, которая базируется на результатах SWOT- и QuaD-анализа, анализе конкурентных технических решений, графике

проведения научного исследования, анализе бюджета научно-технического исследования.

По результатам расчета коэффициента конкурентоспособности научного исследования относительно «WinCC/DataMonitor» компании Siemens получим:

$$K_{kc} = \frac{4,45}{3,85} = 1,15, \text{ а относительно и «Ampla» компании Schneider - } K_{kc} = \frac{4,45}{3,65} =$$

1,22. Эти значения говорят о конкурентоспособности корпоративного веб-портала «SmartReports», поскольку значение коэффициента больше единицы. В результате проведения анализа характеристик разработки посредством технологии QuaD было подсчитано, что средневзвешенное значение показателя качества и перспективности разработки составляет 89,99. Поскольку это значение больше 80, то данная разработка является перспективной. При SWOT-анализе проведен комплексный анализ проекта, выявлены сильные и слабые стороны разработки, а также определены стратегии развития корпоративного портала при возникновении описанных факторов. С точки зрения длительности проведения работ наиболее эффективным вариантом исполнения является исполнение №3. По статье бюджета затрат научно-технического исследования наиболее выгодным является вариант исполнения под номером 2, в соответствии с показателем сравнительной эффективности вариантов исполнения – исполнение № 3.

Глава 5. Социальная ответственность

Занимаясь научно-исследовательской деятельностью, стоит уделить внимание обеспечению безопасности охраны труда и окружающей среды. Такие факторы рассматривает социальная ответственность, а именно: состояние рабочего места, помещения, режим работы, обеспечение мероприятий по защите трудящихся в случае возникновения чрезвычайной ситуации. Существует международный стандарт ИСО 26000 «Руководство по социальной ответственности», разработанный в 2012 г., в соответствии с которым осуществляется разработка системы социальной ответственности на предприятии. При применении организацией данного стандарта для разработки системы социальной ответственности, она решает ряд проблем, таких как: проблемы, связанные с потребителями, трудовые практики, участие в жизни сообществ и их развитие, права человека, окружающая среда, добросовестные деловые практики.

В данной части магистерской диссертации рассмотрен анализ вредных и опасных факторов производства, методы их предупреждения, организация мероприятий защиты в случае возникновения чрезвычайной ситуации. Научно-исследовательская работа представляет собой исследование методов формирования отчетов, а также разработку корпоративного веб-приложения «Smart Reports» с возможностью динамического построения отчетной документации. Следовательно, одним из важнейших критериев исследования является организация рабочего места и режима трудовой деятельности.

5.1 Техногенная безопасность

5.1.1 Отклонения показателей микроклимата

Благоприятные (комфортные) метеорологические условия на производстве являются важным фактором в обеспечении высокой производительности труда и в профилактике заболеваний. При несоблюдении гигиенических норм микроклимата снижается работоспособность человека, возрастает опасность возникновения травм и ряда заболеваний, в том числе профессиональных.

По степени физической тяжести работа программиста относится к категории лёгких работ. В соответствии со временем года и категорией тяжести работ определены параметры микроклимата согласно требованиям [11] и приведены в таблице 17.

Таблица 17. Оптимальные параметры микроклимата рабочего места

Период года	Категория работ по уровням энергозатрат, Вт	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Категория 1а (до 139)	22-24	21-25	40-60	0,1
Теплый		23-25	22-26		0,1

Допустимые величины показателей микроклимата устанавливаются в случаях, когда по технологическим требованиям, техническим и экономически обоснованным причинам не могут быть обеспечены оптимальные величины.

Таблица №18. Допустимые значения микроклимата рабочего стола.

Период года	Категория работ	Температура воздуха, °С		Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
		Ниже опт.	Выше опт.			Ниже опт.	Выше опт.
Холодный	Категория 1а (до 139)	20-21,9	24,2-25	19-26	15-75	0,1	
Теплый		21-22,9	25,1-28	20,29		0,1	0,2

В зимнее время в помещении предусмотрена система отопления. Она обеспечивает достаточное, постоянное и равномерное нагревание воздуха. В соответствии с характеристикой помещения определен расход свежего воздуха согласно [11] и приведен в Таблице 19.

Таблица 19. Нормы подачи свежего воздуха в помещения, где расположены компьютеры.

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, м ³ /на одного человека в час
Объем до 20 м ³ на человека 20...40 м ³ на человека Более 40 м ³ на человека	Не менее 30 Не менее 20 Естественная вентиляция

5.1.2 Недостаточная освещённость рабочей зоны; отсутствие или недостаток естественного света

Производственное освещение — неотъемлемый элемент условий трудовой деятельности человека. При правильно организованном освещении рабочего места обеспечивается сохранность зрения человека и нормальное состояние его нервной системы, а также безопасность в процессе производства. Производительность труда и качество выпускаемой продукции находятся в прямой зависимости от освещения.

Рабочая зона или рабочее место разработчика освещается таким образом, чтобы можно было отчетливо видеть процесс работы, не напрягая зрения, а также исключается прямое попадание лучей источника света в глаза. Кроме того, уровень необходимого освещения определяется степенью точности зрительных работ. Наименьший размер объекта различения составляет 0,5 - 1 мм. В помещении отсутствует естественное освещение. По нормам освещенности [15] и отраслевым нормам, работа за ПК относится к зрительным работам высокой точности для любого типа помещений

Таблица 20. Нормирование освещённости для работы за ПК [7]

1	2	3	4	5	6	7	8	9	10	11
Высокой точности	От 0,3	Б	1	Более 70	300	100	40	15	3,0	1,0
	От 0,5		2	Менее 70	200	75	60	20	2,5	0,7

Где:

- 1-характеристика зрительных работ;
- 2-наименьший или эквивалентный размер объекта различения, мм;
- 3-разряд зрительной работы;
- 4-подразряд зрительной работы;
- 5-относительная продолжительность зрительной работы, %;
- 6-освещенность на рабочей поверхности от системы общего искусственного освещения, лк;
- 7-цилиндрическая освещенность, лк;

- 8-показатель дискомфорта;
- 9-коэффициент пульсации освещенности, %;
- 10-КЕО при верхнем освещении, %;
- 11-КЕО при боковом освещении, %

Требования к освещению на рабочих местах, оборудованных ПК, представлены в таблице 21. [7]

Таблица 21. Требования к освещению на рабочих местах, оборудованных ПК [7]

Освещенность на рабочем столе	300-500 лк
Освещенность на экране ПК	не выше 300 лк
Блики на экране	не выше 40 кд/м ²
Прямая блесккость источника света	200 кд/м ²
Показатель ослеплённости	не более 20
Показатель дискомфорта	не более 15
Отношение яркости:	
– между рабочими поверхностями	3:1–5:1
– между поверхностями стен и оборудования	10:1
Коэффициент пульсации:	не более 5%

Согласно СанПиНу 2.2.1/2.1.1.1278-03 помещения с постоянным пребыванием людей должны иметь естественное освещение. Но помимо естественного освещения, офис должен обладать системой общего освещения. Системы комбинированного освещения рекомендуется применять в помещениях общественных зданий, где выполняется напряженная зрительная работа.

5.1.3 Повышенный уровень шума

Одним из важных факторов, влияющих на качество выполняемой работы, является шум. Шум ухудшает условия труда, оказывая вредное действие на организм человека. Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. Такие нарушения в работе ряда органов и систем организма человека могут вызывать негативные изменения в эмоциональном состоянии человека вплоть до стрессовых. Под воздействием шума снижается концентрация внимания, нарушаются физиологические функции, появляется усталость в связи с повышенными энергетическими затратами и нервно-психическим напряжением, ухуд-

шается речевая коммутация. Все это снижает работоспособность человека и его производительность, качество и безопасность труда. Длительное воздействие интенсивного шума (выше 80 дБ(А)) на слух человека приводит к его частичной или полной потере.

Таблица 22. Предельные уровни звука на рабочих местах (дБ).

Категория напряженности труда	Категория тяжести труда (дБ)			
	1 - легкая	2- средняя	3 - тяжелая	4 – очень тяжелая
1 – мало напряженный	80	80	80	75
2 – умеренно напряженный	70	70	65	65
3 – напряженный	60	60	-	-
4 – очень напряженный	50	50	-	-

При выполнении основной работы на ПЭВМ уровень шума на рабочем месте не должен превышать 50 дБ. Допустимые уровни звукового давления в помещениях для персонала, осуществляющего эксплуатацию ЭВМ при разных значениях частот, приведены в [10].

Для устранения шума необходимо прочистить вентилятор от пыли или заменить полностью.

5.1.4 Повышенный уровень электромагнитных излучений; повышенная напряжённость электрического поля

В данной выпускной работе будет рассматриваться такой источник электромагнитного поля как персональный компьютер.

По [16] энергетическая экспозиция за рабочий день не должна превышать значений, указанных в таблице 23.

Таблица 23. Предельно допустимые значения энергетической экспозиции

Диапазоны частот	Предельно допустимая энергетическая экспозиция		
	По электрической составляющей, $(В/м)^2 \times ч$	По магнитной составляющей, $(А/м)^2 \times ч$	По плотности потока энергии $(мкВт/см^2) \times ч$
30 кГц - 3 МГц	20000,0	200,0	х
3 - 30 МГц	7000,0	Не разработаны	х
30 - 50 МГц	800,0	0,72	х

50 - 300 МГц	800,0	Не разработаны	x
300 МГц - 300 ГГц	x	x	200,0

Для обеспечения меньшего электромагнитного излучения использован жидкокристаллический монитор. Необходимо учитывать расстояние до монитора, так как при большем расстоянии от человека оказывается меньшее влияние. В связи с тем, что электромагнитное излучение от стенок монитора намного больше, необходимо ограничивать его стенами, т.е. ставить в углу. Необходимо чтобы компьютер был заземлен, а также необходимо по возможности сокращать время работы за компьютером.

5.2 Электробезопасность

Электробезопасность – система организационных и технических мероприятий и средств, обеспечивающих защиту людей от вредного и опасного воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

При работе с компьютером существует опасность электропоражения:

при прикосновении к нетоковедущим частям, оказавшимся под напряжением (в случае нарушения изоляции токоведущих частей ПЭВМ);

при прикосновении с полом, стенами оказавшимися под напряжением;

имеется опасность короткого замыкания в высоковольтных блоках: блоке питания и блоке дисплейной развертки.

В зависимости от условий в помещении опасность поражения человека электрическим током увеличивается или уменьшается. По [12] помещение, в котором находится рабочее место, относится к категории помещений без повышенной опасности. Его можно охарактеризовать, как сухое, непыльное, с токонепроводящими полами и нормальной температурой воздуха. Температурный режим, влажность воздуха, химическая среда не способствуют разрушению изоляции электрооборудования.

Безопасность при работе с электроустановками обеспечивается применением различных технических и организационных мер. Основные технические средства защиты от поражения электрическим током:

- изоляция токопроводящих частей и ее непрерывный контроль;
- установка оградительных устройств;
- предупредительная сигнализация и блокировки;
- защитное заземление;
- зануление;
- защитное отключение по [12].

5.3 Региональная безопасность

В результате осуществления рабочей деятельности возникает потребность в утилизации электронной техники и комплектующих: системный блок, монитор, принтер и т.п. При наличии современных средств утилизации стало возможным перерабатывать 95% отходов и всего 5% отправить на свалки. При переработке используется ручной и автоматизированный труд в соотношении 50/50. Все работы по утилизации производятся строго с применением средств защиты организма (защита органов дыхания, спецодежда, перчатки и т. д.).

На данный момент государство активно исполняет программу экономии энергоресурсов, что означает использование люминесцентных ламп вместо ламп накаливания. Также достоинством использования ламп данного типа является мягкое и равномерное распределение света. Однако, по причине содержания вредных веществ (ртуть, амальгама, аргон, и т.п.) данный источник света составляет угрозу загрязнения окружающей среды. Существует множество способов утилизации. Первый – гидрометаллургический (жидкофазный) способ демеркуризации, который подразумевает мокрое измельчение ламп с одновременной отмывкой ртути в два подхода. Второй – термическая демеркуризация. В основе данной технологии лежит измельчение ламп, нагрев полученного стеклобоя с целью перевода ртути в газообразное состояние и удаления технологического газа, переработка очищенного стеклобоя. Данная технология позволяет удалить из стеклобоя до 95% люминофор.

5.4 Организационные мероприятия обеспечения безопасности

5.4.1 Эргономические требования к рабочему месту

К элементам рабочего места следует отнести стол и кресло или стул. Основное положение при работе – положение сидя, т.к. снижает риск возникновения утомления работника. Рекомендуемые параметры сидения: высота должна быть в пределах 420-550мм, поверхность – мягкая, угол наклона спинки – регулируемый.

Существуют требования по отношению к рабочей позе работника. Основные аспекты: голова не должна быть наклонена более чем на 20° , плечи – расслаблены, локти находятся в положении под углом 80° - 100° , предплечья и кисти рук должны в горизонтальном положении.

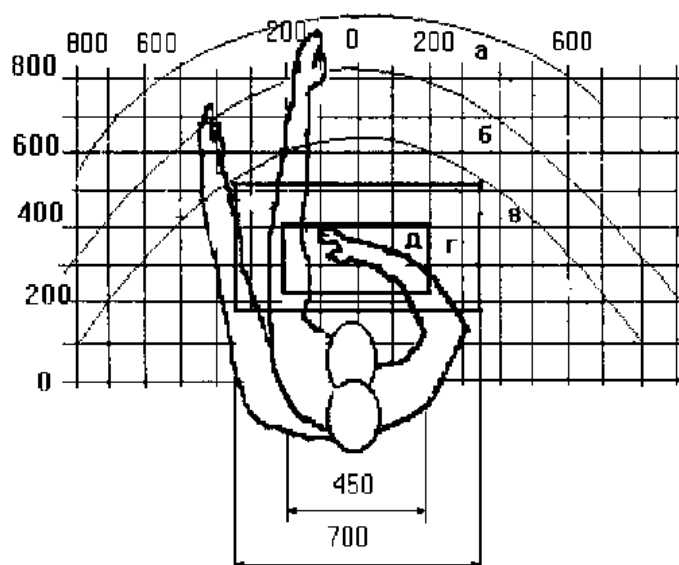


Рис. 29. Зоны досягаемости рук в горизонтальной плоскости.

а – зона максимальной досягаемости

б – зона досягаемости пальцев при вытянутой руке

в – зона легкой досягаемости ладони

г – оптимальное пространство для грубой ручной работы

д – оптимальное пространство для тонкой мужской работы

Оптимальное размещение предметов труда и документации в зонах досягаемости:

– дисплей размещается в зоне а(в центре);

- системный блок размещается в предусмотренной нише стола;
- клавиатура – в зоне z/d ;
- «мышь» – в зоне справа;
- сканер в зоне a/b (слева);
- принтер находится в зоне a (справа);
- документация, необходимая при работе – в зоне легкой досягаемости ладони – e , а в выдвижных ящиках стола – литература, неиспользуемая постоянно.

На рисунке 30 показан пример размещения основных и периферийных составляющих ПК на рабочем столе программиста.

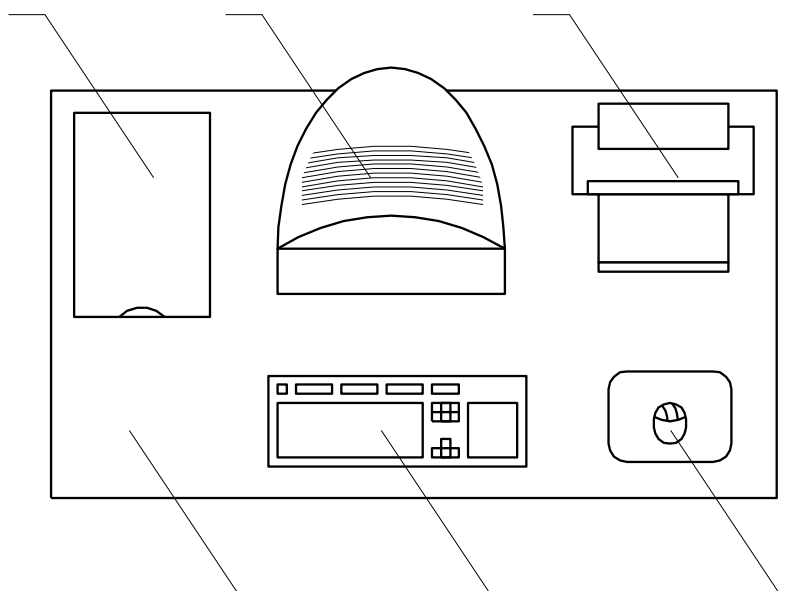


Рис. 30. Размещение основных и периферийных составляющих ПК.

- 1 – сканер, 2 – монитор, 3 – принтер, 4 – поверхность рабочего стола,
5 – клавиатура, 6 – манипулятор типа «мышь».

Для комфортной работы эргономика рабочего пространства должна удовлетворять следующим требованиям [8].

5.4.2 Окраска и коэффициенты отражения

В зависимости от ориентации окон рекомендуется следующая окраска стен и пола:

– окна ориентированы на юг – стены зеленовато-голубого или светло-голубого цвета, пол – зеленый;

– окна ориентированы на север – стены светло-оранжевого или оранжево-желтого цвета, пол – красновато-оранжевый;

– окна ориентированы на восток – стены желто-зеленого цвета, пол зеленый или красновато-оранжевый;

– окна ориентированы на запад – стены желто-зеленого или голубовато-зеленого цвета, пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения для потолка 60-70%, для стен 40-50%, для пола около 30%.

5.5 Особенности законодательного регулирования проектных решений

Государственный надзор и контроль в организациях независимо от организационно-правовых форм и форм собственности осуществляют специально уполномоченные на то государственные органы и инспекции в соответствии с федеральными законами.

К таким органам относятся Федеральная инспекция труда, Государственная экспертиза условий труда Федеральная служба по труду и занятости населения (Минтруда России Федеральная служба по экологическому, технологическому и атомному надзору (Госгортехнадзор, Госэнергонадзор, Госатомнадзор России) Федеральная служба по надзору в сфере защиты прав потребителей и благополучия человека (Госсанэпиднадзор России) и др.

Так же в стране функционирует Единая государственная система предупреждения и ликвидации чрезвычайных ситуаций (РСЧС), положение о которой утверждено Постановлением Правительства Российской Федерации [13], в соответствии с которым, система объединяет органы управления, силы и средства.

Продолжительность рабочего дня не должна превышать 40 часов в неделю.

Не рекомендуется работать за компьютером более 6 часов за смену;

- рекомендуется делать перерывы в работе за ПК продолжительностью 10 минут через каждые 50 минут работы;
- продолжительность непрерывной работы за компьютером без регламентированного перерыва не должна превышать 2 часов;
- во время регламентированных перерывов целесообразно выполнять комплексы упражнений.
- при нерегламентированной работе повышенной интенсивности возможны головные боли, нервные срывы и др.

5.6 Пожарная безопасность

Пожарная безопасность может быть обеспечена мерами пожарной профилактики и активной пожарной защиты. Пожарная профилактика включает комплекс мероприятий, направленных на предупреждение пожара или уменьшение его последствий. Активная пожарная защита – меры, обеспечивающие успешную борьбу с пожарами или взрывоопасной ситуацией.

Возникновение пожара в помещении, где установлена вычислительная и оргтехника, приводит к большим материальным потерям и возникновению чрезвычайной ситуации. Чрезвычайные ситуации приводят к полной потере информации и большим трудностям восстановления всей информации в полном объёме.

Согласно нормам технологического проектирования [14], в зависимости от характеристики используемых в производстве веществ и их количества, по пожарной и взрывной опасности помещения подразделяются на категории А, Б, В, Г, Д.

Данное помещение относится к категории В [15], производства, связанные с обработкой или применением твердых сгораемых веществ и материалов.

Для исключения возникновения пожара необходимо:

- вовремя выявлять и устранять неисправности;
- не использовать открытые обогревательные приборы, приборы кустарного производства в помещении лаборатории;

– определить порядок и сроки прохождения противопожарного инструктажа и занятий по пожарно-техническому минимуму, а также назначить ответственного за их проведения.

В случае возникновения пожара необходимо отключить электропитание, вызвать по телефону пожарную команду, произвести эвакуацию и приступить к ликвидации пожара огнетушителями. При наличии небольшого очага пламени можно воспользоваться подручными средствами с целью прекращения доступа воздуха к объекту возгорания.

Для тушения пожаров в помещении необходимо установить углекислотный огнетушитель типа ОУ-5.

Покидать помещение согласно плану эвакуации на рисунке 31.

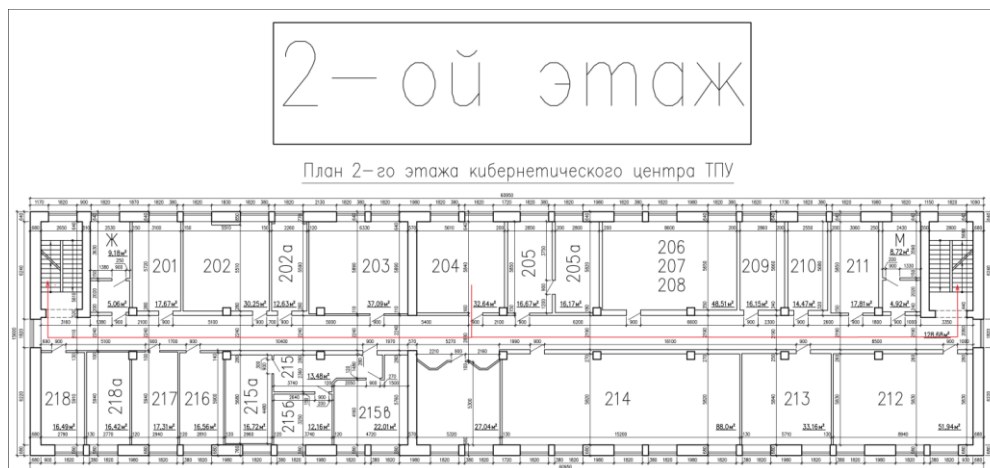


Рис. 31. План эвакуации при пожаре и других ЧС из помещения учебного корпуса Кибернетического Центра ТПУ, ул. Советская 84/3, 2 этаж.

Список публикаций

1. МСИТ 2016 - «Обзор основных современных технологий разработки web-приложений».
2. МСИТ 2017 – «Разработка интерактивной системы обработки, анализа и управления технологическими данными».

Список использованных источников

1. Node.js. Путеводитель по технологии. Кирилл Сухов. – М.: ДМК Пресс, 2015. – 416 с.
2. Node.js. Путеводитель по технологии. – М.: ДМК Пресс, 2015. – 416 с.
3. Rails 4. Гибкая разработка веб-приложений. — СПб.: Питер, 2014. — 448 с.
4. ASP.NET MVC Framework с примерами на C#. Сандерсон Стивен – Вильямс, APress, 2010. – 557 с.
5. <https://professorweb.ru/my/entity-framework/6/level1/>
6. <https://metanit.com/sharp/entityframework/1.1.php>
7. СП 52.13330.2011 Свод правил. Естественное и искусственное освещение.
8. СанПиН 2.2.2/2.4.1340 – 03. Санитарно-эпидемиологические правила и нормативы «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы». – М.: Госкомсанэпиднадзор, 2003.
9. СанПиН 2.2.1/2.1.1.1278 – 03. Гигиенические требования к естественному, искусственному и совмещённому освещению жилых и общественных зданий. М.: Минздрав России, 2003.
10. СН 2.2.4/2.1.8.562 – 96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки.
11. СанПиН 2.2.4.548 – 96. Гигиенические требования к микроклимату производственных помещений. М.: Минздрав России, 1997.
12. ГОСТ ИЕС 60950-1-2014 Оборудование информационных технологий. Требования безопасности.
13. Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ.
14. НПБ 105-03. Нормы пожарной безопасности. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности.

15. Технический регламент «о требованиях пожарной безопасности» [Электронный ресурс]: Единая справочная служба Консорциума «Кодекс». – Режим доступа: свободный. Ссылка доступа: <http://ezproxy.ha.tpu.ru:2065/docs/>
16. СНиП 23-05-95 Естественное и искусственное освещение, 2010.
17. СанПиН 2.2.4.1191-03 «Электромагнитные поля в производственных условиях»
18. www.tutorialspoint.com/nodejs
19. Node.js in Action/ Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich// Manning Publications Company. – 2013. – Vol 416.
20. Agile Web Development with Rails/ Sam Ruby, Dave Thomas, David Heinemeier Hansson// The Pragmatic Bookshelf. – 2012. – Vol. 476.
21. Pro ASP.NET MVC 5/ Adam Freeman// Apress. – 2012. – Vol. 812 .
22. C# 6.0 and the .NET 4.6 Gramework/ Andrew Troelsen, Philip Japikse// Apress. – 2005. – Vol. 1625.
23. Programming ASP.NET MVC 4/Jess Chadwick, Todd Snyder, and Hrusikesh Panda// O'Reilly. – 2012. – Vol. 471.

Заключение

В результате выполнения данной магистерской диссертации были разработаны алгоритмы обработки данных и динамического формирования отчетов, проведено исследование вариантов реализации веб систем с использованием разработанных алгоритмов. Рассмотрены два подхода, наиболее эффективным, гибким и функциональным оказался метод динамического построения алгоритма пользователем. Данный подход требует меньших временных затрат для разработки, а также облегчает сопровождение алгоритма построения отчета. Для реализации данной методологии использовалась платформа ASP.NET MVC на языке C#. Выбор данной технологии обосновано в разделе литературного обзора.

Программная реализация разработанных алгоритмов была выполнена в рамках создания большого корпоративного веб-портала «SmartReports» компании «Элеси». Для хранения конфигурации веб-портала была разработана схема базы данных, реализованная на СУБД PostgreSQL. Для осуществления связи веб-портала с базой данных применялась технология EntityFramework, позволяющая придерживаться объектно-ориентированного подхода на протяжении всей разработки веб-приложения.

Для реализации интуитивно понятного интерфейса применялись front-end графические библиотеки, позволяющие создать интерфейс в стиле Windows 8.

Приложение А (Информативное)

Web Applications Development Technologies Review

Node.js Technology

Node.js is a server-side open-source platform built on JavaScript Engine (V8 Engine) from Google. It was developed in 2009 by Dahl. It allows to create fast and scalable network applications. Node.js uses JavaScript programming language. It is cross-platform application and can be run on OS X, Microsoft Windows, and Linux.

Node.js has a number of features [18]:

- Asynchronous events. All the API libraries, that Node.js uses, are asynchronous and non-blocking. Thus, server based on Node.js does not wait for data to return from API.
- Fast execution. As Node.js is built on Google Chrome's V8 JavaScript Engine, the Node.js code executes very fast.
- Single threading and highly scalable. Server built on Node.js uses a single threaded model and can respond in a non-blocking manner. It can provide a larger number of requests than a traditional Apache HTTP Server.
- Node.js applications do not buffer data.

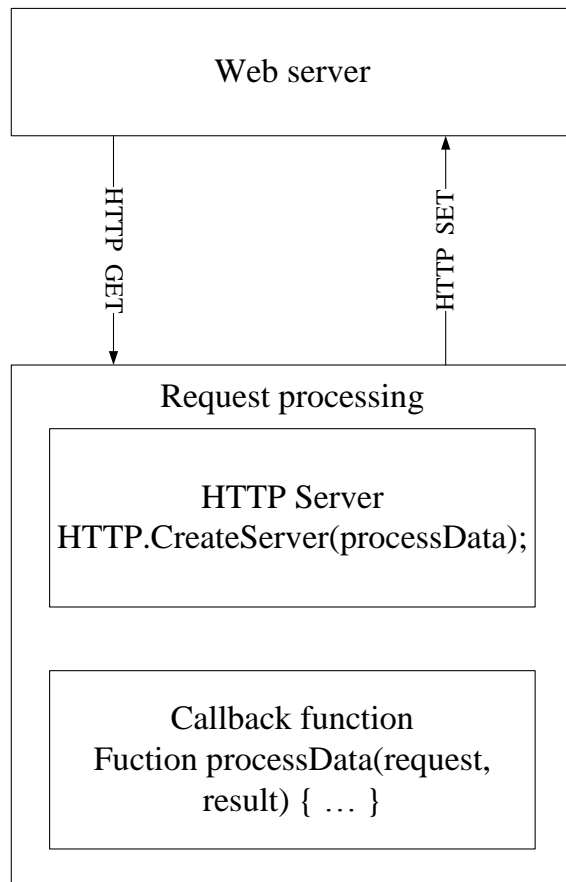
Asynchronous events.

While addressing to the application, Node does not create a new program flow or process for each request, but listens to described events. If even happened, Node web-server executes a specific script. Web-server Node.js handles events according to FIFO (First-In-First-Out) with asynchronous execution of events, that allows to avoid blocking of requests to the server. Web-application, that was written on Node.js, is a single threading. It is asynchronous nature of Node.js technology that makes possible to process a huge number of interactions at web-page and allows to create a responsive interface, that does not make someone to wait for connection to server resources.

Node web applications development peculiarities

Node.js supports HTTP interface using `http` module. To create HTTP server create function `http.createServer()`. This function has the only argument – a callback function, that is called by each HTTP request, that the server accepts.

Each HTTP request, that the server receives, calls the callback function. Before executing callback function Node executes parsing of the query until the end of the HTTP heading. Node cannot start parsing the body of a query until the callback function will be executed. This behavior is different from server-based development framework, such as PHP, that parses heading and body of http request before application code execution. Node suggests low-level interface, that enables to process request body as it parses.



Picture 32. HTTP request lifetime in Node.js application with HTTP server.

Picture 32 describes HTTP request lifetime. HTTP client (web browser) makes an HTTP request to the application. Node application accepts connection and data in the request. HTTP server parses request and calls the callback function (`processData`). The function executes its algorithm and returns a response. After that the request re-

turns to the web browser, that forms an appropriate HTTP-response for the client [19, p. 111]..

RESTful Interface

Node.js supports API REST interface (Representational State Transfer). This concept was suggested by Roy Fielding in 2000. He is one of the cofounders of the HTTP 1.0 and HTTP 1.1 protocols specifications. According to this paradigm there are HTTP verbs, such as GET, POST, PUT and DELETE. To create REST-adaptive server it is required to implement these HTTP nouns. Each noun plays its strongly defined role:

- GET – gets the list of requested elements;
- POST – adds element to the resource;
- DELETE – deletes element from the resource;
- PUT – updates existing element.

The program code, that implements the HTTP-request processing, is listed below.

```
var http = require('http')
var server = http.createServer(function(req, res){
  switch (req.method){
    case 'GET':
      //извлечение элементов из массива
      items.forEach(function(item,i){
        //действия над извлеченными элементами
        res.end();});
      break;
    case 'POST':
      //перемещение фрагментов в массив
      var item = '';
      req.on('data', function(chunk){
        item+=chunk; });
      req.on('end', function(){
        items.push(item);
        res.end();});
      break;
    case 'DELETE':
      var urlPath = url.parse(req.url).pathname;
      var i = parseInt(urlPath.slice(1),10);
      if(isNaN(i)){
        //проверка корректности числа
        res.statusCode = 400;
        res.end('Invalid item id');}
      else if(!items[i]){
        // проверка существования элемента в массиве по запрашиваемым
        индексом
        res.statusCode = 404;
        res.end('Item not found');}
      else{
```



```

        //Удаление запрошенного элемента
        items.splice(i,1);
        res.end('OK\n');}
    break; }
case 'PUT':
    var urlPath = url.parse(req.url).pathname;
    var i = parseInt(urlPath.slice(1),10);
    if(isNaN(i)){
        //является ли i числом
        res.statusCode = 400;
        res.end('Invalid item id');}
    else if(!items[i]){
        // проверка существования элемента в массиве по запрашиваемым
индексом
        res.statusCode = 404;
        res.end('Item not found');}
    else{
        //Редактирование запрошенного элемента
        items[i].name = req.body.name;
        items[i].description = req.body.description;
        res.end('OK\n');}
    break; } }

```

Node.js checks used HTTP method by means of “req.method” property. As the method was defined, its behavior is described. When the HTTP code parsing system in Node reads and parses request data, they become available in the format of data and contain fragments of the parsed data that are ready for further processing. The “req.end” event marks the end of the request. After the “statusCode” object “res” enables to handle the server exception by setting the value to the property [19, p. 114].

MVC design pattern of Node.js web applications

Node.js allows to implement MVC architecture (Model-View-Controller). This architecture became popular because of web applications growing popularity. Using this pattern it is possible to create structured code in order to delimit application infrastructure.

Model represents data and methods to process the data. It changes its condition depending on request. Model does not contain information about the data, that a user will receive. View is responsible for information depicting (data visualization). Controller provides the connection between the user and the server part of web application. It manages data input, data flows, transforms the model. It is significant that model does not depend on visualization.

Node applications, that implement MVC architecture, are able to restrict the model and controller level, which are the server and client parts of web application. Thus, model and controller operating at the server and view operates in web browser.

Node application testing

As web applications functionality expands, the risk of error embedding increases. The popularity of computer-aided testing is steadily growing. This technology allows to describe the logic of testing by a programmer. There are two basic types of computer-aided testing: unit tests and acceptance test.

Module tests check the logic of an application functionality. This type of computer-aided testing enables to check separate parts of web application and to introduce bugs at the early stage of project development. Using module tests there is a warranty, that current changes will not lead to new bugs occurrence. Acceptance tests is applied while web applications debugging and controls scenarios, that are accepted by web browser. It check web applications functionality to [19, p. 357].

Ruby on Rails Technology

After the Ruby on Rails debut, the framework became one of the most powerful and popular frameworks for dynamic web applications development. Ruby on Rails is in the open source and has the MIT license. Consequently, it is possible to use it freely. The result of such a popularity of the Rails framework is due to its elegant and compact design. Using the flexibility of Ruby, the programming language Ruby on Rails is based on, it creates domain-specific language. As a result, there are a lot of common tasks of web programming such as HTML generation, data models creation and URI routing. All these components are easy to implement in Rails application. The result code of web application is short and readable [20, p.16].

Rails application are developed on Ruby – a contemporary domain-specific language of scenarios. The programming language allows to create succinct and easy-to-read code.

All the Rails applications implement MVC architecture. The role of model in Ruby on rails is the same as in other technologies that implement MVC architecture. Model is a container that works with databases. Within this technology tables, de-

scribed in database, are models in the application and table fields are model fields respectively. But this approach troubles compatibility between database and object-oriented programming language, because objects are described by data and operations and database is described by value sets. Operations that are easily expressed by means of relation concepts are sometimes hardly programmed in object-oriented systems. The inverse proportion is also right [20, p.17].

To provide connection between the model and the database object-relational mapping is used. Thus, if a database contains table “Book”, Ruby on Rails project contains class with the same name.

Active Record is an ORM-insertion provided by Rails. It follows ORM model standards: tables are described as classes, rows as objects and columns as properties. It differs from the most other libraries by configuration method. Based on agreements and optimal settings by default Active Record minimizes settings for a programmer.

The code listed below selects a book with identifying number equals to 1, changes its name and saves changes [20, p. 55].

```
require 'active_record'
class Book < ActiveRecord::Base
end
book = Book.find(1)
book.name = "Война и мир"
book.save
```

View and controller in MVC architecture are closely related to each other. According to this reason Rails developers decided to combine them into a single component Action Pack with a clear logics separation.

View in Rails is responsible for creation and partial response, depicted in web browser, processed by application or sent as email. The simplest form of view is a fragment of HTML code.

Dynamic content in Rails generates by three types of templates. According to the most popular scheme of templates creation called Embedded Ruby (ERb): Ruby code fragments are inserted in the view, which is similar to the method, applied in other web environments such as PHP and JSP. Embedding code into view we take risks to implement application logics in it, which must be in model or in a controller.

ERb might be used to construct JavaScript fragments on a server which are executed in web browser. This technology perfectly fits AJAX dynamic interfaces creation.

Rails has XML Builder tool that allows constructing XML documents, which use Ruby code. The structure of generated XML document will be automatically followed by code structure.

Rails controller is a logical center of web application. It coordinates cooperation between user, view and model. Controller is responsible for several important functions [20, p. 56]:

- redirection of external requests to internal actions;
- caching control, that accelerates application in several times;
- helper module control, that extends view templates possibilities without code size increase;
- session control, that arises sensation of continuous application functioning.

Some suppose that Ruby on Rails is the most sophisticated software development technology, because it already contains all the necessary modules.

ASP.NET Technology

At the end of 1990s ASP.NET Framework appeared. It became the continuer of such technologies as ASP (Active Server Pages) and JSP (Java Servlet Pages). ASP.NET is a set of technologies for web applications development, that work under IIS web server (Internet Information Services).

ASP.NET Web Forms

ASP.NET Web Forms is a web applications development technology based on object-oriented approach with form-based templates support. Web Forms files are server controls containers containig in System.Web namespace and having file extension .aspx. There are files containig application business logic in Web Forms project [21, p. 2].

Web Forms are similar to Windows Forms, because server controls may display data and initiate events, on which handlers may be attached.

ASP.NET Web Forms model was initially developed implement RAD (Rapid Application Development) potential. Thus, the main determining factor for the majority of basic characteristics and key concept of ASP.NET was productive programming. Web Forms model is based on three basic concepts: page postback, view state and server controls. Each HTTP request accepted by the server and associated with runtime environment ASP.NET goes through several stages, the most important of which is postback event handling. Postback Event is the main action, which is expected be the user as a result of query processing [22, p. 1397].

At first the request is processed to extract heading information, which is necessary for successful postback. This information contains the state of server controls, which collaboratively form the final HTML markup. After the postback HTML response is generated for the browser with new server controls state, which will be used in the next request.

ASP.NET web page is based on one form component containing all the input elements, that user may interact with. The form also contains controls for sending data (buttons and links).

View state is a dictionary used by ASP.NET web pages to store child control state between postbacks [21, p. 6]. View state plays a key role in implementing postback model. Without the use of view state it would be impossible to track the state of ASP.NET web pages. View state is a unique and encrypted hidden field, which stores the dictionary of controls values of a unique ASP.NET web page form. Automatically each web page control saves its state and properties values in view state. These data are sent to the client and are accepted by the server with each web page request. Nevertheless they are never used on a client side of a server.

At the first stages of ASP.NET Web Forms development the contact between HTML and JAVA script code was a benefit of the technology. But with a rapid growth of AJAX has changed the concept of web applications possibilities, and more importantly, has considerably changed user's expectations. As a result, web applications require being more interactive. One of the methods of response acceleration is

the increase of code size, which is executed in web browser only if a certain page is showed.

ASP.NET MVC

Microsoft introduced ASP.NET MVC as an alternative to Web Forms. Such separation of an application into view, model and controller allows to improve unit testing and as a result to advance the control over the web application. The main problem of Web Forms is the transfer of a huge amount of data in view state, which is fixed in MVC-based application.

As in other technologies, MVC applications are based on models, view and controllers. Model is a data container. Controller processes client requests and returns results of these requests. View shows data of the model.

MVC solution has following structure: “Models”, “Views”, “Controllers” folders contain model, view and controllers classes respectively; “Scripts” folder contains client JavaScript files; “Default.aspx” file contains routing table associating requests to the web application with methods in controllers; “Web.config” file contains web application description, including connection strings to databases.

Query lifetime in ASP.NET MVC application is very simple. Client (web browser) sends an HTTP request, which is processed by routing table. It defines the name of controller and its action for processing the request. In terms of these request parameters controller method (action) executes a certain algorithm generating view for showing in web browser [23, p.16].

The main link between controller and view is a View Data collection that enables exchanging data between these components of web application.

As the markup was generated by the view engine, web server sends it as a response upon the HTTP protocol. This is the last stage of MVC application request lifetime.

MVC framework allows to transform URL requests into controller action using ASP.NET routing. “Global.asax” file includes RegisterRoutes method, which contains the following operator [23, p.17]:

```
routes.MapRoute( "Default", "{controller}/{action}/{id}", new { controller =  
"Home", action = "Index", id = UrlParameter.Optional } );
```

This operator calls the `MapRoute` method to register route. Firstly, it defines the query format so as backlink to the MVC application to be returned in a Controller/Action/Identifier format (“{controller}/{action}/{id}”). Another important part of the operator is the last method argument `MapRoute`, that specifies the default value.

Conclusion

Due to distinctions between ASP.NET MVC Framework and Ruby on Rails developers, who formerly worked on .NET framework will easily cope with ASP.NET MVC framework while developers preferred Python or Ruby programming languages will easily start working with Ruby on Rails. If Ruby on Rails has a full stack of technologies to work with databases, .NET has a huge amount of such technologies, among which developer is free to choose. In .NET framework these components will not be closely related to each other as in Ruby on Rails.

By comparison ASP.NET and Node.js it is clear that the choice between these technologies depends on the developer preferences, because both of them may use external libraries and have sufficient productivity, implement asynchronous queries processing and MVC architecture.

Приложение Б
(Информативное)
Программный код интерфейса IBlock

```
/// <summary>
/// Интерфейс для реализации блоков функциональной схемы
/// </summary>
public interface IBlock : ISchemeItem, IClonable<IBlock>
{
    /// <summary>
    /// Порядок выполнения алгоритма в функциональной схеме
    /// </summary>
    int ExecutionOrder { get; set; }

    /// <summary>
    /// Комментарий к блоку
    /// </summary>
    string Comment { get; set; }

    /// <summary>
    /// Выполнить действие (алгоритм) функционального блока
    /// </summary>
    void Execute();
}
```


Приложение В
(Информативное)
Программный код реализации блока «Сумма»

```
public class Sum : IBlock
{
    public Input<int> In1 { get; set; }

    public Input<int> In2 { get; set; }

    public Output<int> Out { get; set; }
    public Input<List<IPin>> ArrayGroup { get; set; }

    public Sum()
    {
        In1 = new Input<int>(this);
        In1.PossibleTypes.Add(typeof(int));
        In2 = new Input<int>(this);
        In2.PossibleTypes.Add(typeof(int));
        ArrayGroup = new Input<List<IPin>>(this) { Description = "Массивы" };
        ArrayGroup.PossibleTypes.Add(typeof(DataTable));
        ArrayGroup.Value = new List<IPin>
        {
            new Input<DataTable>(this, ArrayGroup) {Description = "In13", Value = new
DataTable()},
        };

        Out = new Output<int>(this);
        Out.PossibleTypes.Add(typeof(int));

        Category = "Арифметические";
        Description = "Сумма";
        Comment = "Сумма";
    }

    public override void Execute()
    {
        Out.Value = In1.Value + In2.Value;
    }
}
```

Приложение Г (Информативное)

Программный код реализации интерфейса `IDataSource`

```
/// <summary>
/// Интерфейс для реализации источников данных
/// </summary>
public interface IDataSource: IUnique
{
    /// <summary>
    /// Конфигурация
    /// </summary>
    Configuration.Configuration Configuration { get; set; }

    /// <summary>
    /// Параметры источника данных
    /// </summary>
    DataSourceParams DataSourceParams { get; set; }

    /// <summary>
    /// Возвращает состояние сервера
    /// </summary>
    bool IsConnected { get; }

    /// <summary>
    /// Подключение к серверу
    /// </summary>
    void Connect();

    /// <summary>
    /// Отключение от сервера
    /// </summary>
    void Disconnect();
}
```

Приложение Д (Информативное)

Программный код реализации интерфейса IPipe

```
/// <summary>
/// Интерфейс для реализации связей между функциональными блоками
/// </summary>
public interface IPipe : ISchemeItem, IDisposable
{
    Type ContentType { get; set; }

    IPin GetInput();

    IPin GetOutput();
}

/// <summary>
/// Интерфейс для реализации связей с передаваемыми данными между функциональными блоками
/// </summary>
/// <typeparam name="T">Тип передаваемых данных между функциональными блоками</typeparam>
public interface IContentPipe<T> : IPipe
{
    /// <summary>
    /// Вход данной связи
    /// </summary>
    Output<T> In { get; set; }

    /// <summary>
    /// Выход данной связи
    /// </summary>
    Input<T> Out { get; set; }
}
```