МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

**"Южно-Уральский государственный университет"**

**(национальный исследовательский университет)**

**Факультет Вычислительной математики и информатики**

**Кафедра системного программирования**

**Разработка веб-приложения интернет-магазина для ООО «Фантазия»**

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

ЮУрГУ – 010400.62.2013.09-163-1705.КР

|  |  |
| --- | --- |
| Нормоконтролер,доц. каф. СП  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.И. Радченко  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2013 г. | Научный руководитель:  к.ф.-м.н.  \_\_\_\_\_\_\_\_\_\_Г.И. Радченко  Автор работы:  студент группы ВМИ-346  \_\_\_\_\_\_\_\_\_\_ Е.Е. Абзалов  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |

Челябинск 2013

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

**«Южно-Уральский государственный университет»**

**(национальный исследовательский университет)**

**Факультет Вычислительной математики и информатики**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

02.10.2013

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы ВМИ-346 Е.Е.Абазлову,

обучающемуся по направлению 010400.62 «Информационные технологии»

1. **Тема работы**Разработка веб-приложения интернет-магазина для ООО «Фантазия»
2. **Срок сдачи студентом законченной работы:** 26.12.2013 г.
3. **Исходные данные к работе (три основных источника)**
4. Котеров Д., Костарев А. PHP 5 Наиболее полное руководство в подлиннике (2-е издание). – БХВ Петербург, 2008. – 1107 с.;
5. Myer T. Professional CodeIgniter – Wiley Publishing, Inc, 2008. – 339 с.;
6. Darie C., Balanescu E. Beginning PHP and MySQL E-Commerce from novice to professional (2 edition) – Apress, 2008. -737 с.
7. **Перечень подлежащих разработке вопросов**
8. Изучить технологии программирования PHP;
9. Изучить подходы к разработке веб-приложений электоронной коммерции;
10. Изучить возможности фреймворка CodeIgniter;
11. Реализовать веб-приложение интернет-магазина;
12. Произвести тестирование веб-приложения.

**Дата выдачи задания:** «2» октября 2013 г.

Научный руководитель Г.И. Радченко

Задание принял к исполнению Е.Е. Абзалов

# Оглавление

[Оглавление 3](#_Toc357779415)

[Введение 4](#_Toc357779416)

[Актуальность работы 4](#_Toc357779417)

[Цели и задачи работы 4](#_Toc357779418)

[Структура и объем работы 5](#_Toc357779419)

[1. Обзор технологий построения информационных систем 7](#_Toc357779420)

[1.1 Технологии проектирования информационных систем 7](#_Toc357779421)

[1.2 СУБД для информационной системы 8](#_Toc357779422)

[1.3 Архитектура клиент-сервер 9](#_Toc357779423)

[1.4 Варианты построения информационных систем с веб-интерфейсом 10](#_Toc357779424)

[1.5 Вывод 11](#_Toc357779425)

[2. Требования к веб-приложению биллинговой системы 13](#_Toc357779426)

[2.1 Функциональные требования 13](#_Toc357779427)

[2.2 Нефункциональные требования 14](#_Toc357779428)

[2.3 Прецеденты использования биллиноговой системы 14](#_Toc357779429)

[2.4 Вывод 16](#_Toc357779430)

[3. Архитектура веб-приложения биллинговой системы 17](#_Toc357779431)

[3.1 Общий интерфейс системы 17](#_Toc357779432)

[3.2 Интерфейс для работника договорного отдела 17](#_Toc357779433)

[3.3 Интерфейс для экономиста 18](#_Toc357779434)

[3.4 Интерфейс для расчетчика 19](#_Toc357779435)

[3.5 Интерфейс для сотрудника по обработке оплат 19](#_Toc357779436)

[3.6 Интерфейс для специалиста по работе с задолженностью 20](#_Toc357779437)

[3.7 Бизнес-аналитик 21](#_Toc357779438)

[3.8 Вывод 21](#_Toc357779439)

[4. Реализация веб-приложения биллинговой системы 22](#_Toc357779440)

[4.1 Схема базы данных 22](#_Toc357779441)

[4.2 Файловая схема проекта 23](#_Toc357779442)

[4.3 Организация безопасности приложения 25](#_Toc357779443)

[5. Тестирование веб-приложения биллинговой системы 30](#_Toc357779444)

[5.1 Тестирование системы безопасности приложения 30](#_Toc357779445)

[5.2 Тестирование пользовательского интерфейса системы 32](#_Toc357779446)

[Заключение 34](#_Toc357779447)

[Литература 35](#_Toc357779448)

[Приложение 1. Описание прецедентов 36](#_Toc357779449)

# Введение

## Актуальность работы

В процессе работы энергосбытовой компании необходимо хранить и обрабатывать большие объемы информации – данные о потребителях, договорах, начислениях, оплатах задолженности. Вся эта информация должна храниться в структурированной базе данных, для обеспечения ее автоматизированной обработки. При этом необходимо обеспечить возможность удобного добавления, изменения или анализа информации из такой базы данных сразу несколькими пользователями. Для этого необходимо использовать биллинговую систему.

Существующая биллинговая система построена по технологии клиент-сервер, где пользовательское приложение представляет собой классическое «настольное» приложение, непосредственно взаимодействующее с базой данных в процессе работы. На данный момент такой подход начинает устаревать из-за неудобства сопровождения и ограничений по масштабируемости, включая:

1. сложность одновременного обновления всех клиентских приложений при незначительных изменениях бизнес-логики;
2. высокие требования к масштабируемости сервера баз данных, которому необходимо поддерживать отдельное соединение для каждого работающего с ним клиента;
3. сложность поддержки и конфигурирования рабочих станций.

В связи с этим является актуальной задача модернизации существующей клиент-серверной биллинговой системы и разработка веб-приложения для работы с биллинговой системой энергосбытовой компании.

## Цели и задачи работы

Основной целью данной работы является разработка веб-приложения для работы с базой данных энергосбытовой компании. Приложение предназначено для автоматизации деятельности энергосбытовой компании в области биллинга. Оно должно обеспечивать ведение реестра контрагентов и договоров энергоснабжения, выполнение начислений за потребленную электроэнергию, учет поступившей оплаты, расчет дебиторской задолженности. Веб-приложение будет использоваться для формирования различных отчетов, которые необходимы для принятия управленческих решений.

Для достижения цели надо решить задачи:

1. изучить возможности работы с базой данных Oracle Database 11g;
2. изучить возможности работы с сервером приложений OC4J;
3. изучить существующие подходы разработки веб-приложений для работы с базой данных;
4. изучить технологии программирования Java EE, Oracle ADF;
5. изучить среду разработки веб-приложений Oracle JDeveloper 11g;
6. реализовать веб-приложение для работы с биллинговой системой энергосбытовой компании.

## Структура и объем работы

Работа состоит из введения, пяти глав, заключения, библиографии и приложения. Объем работы составляет 44 страницы, объем библиографии – 14 источников.

В первой главе дается обзор технологий построения информационных систем. В этом разделе будут рассмотрены технологии проектирования и построения информационных систем и выявлены наиболее перспективные из них.

Вторая глава содержит описание и анализ требований к информационной системе.

В третьей главе представлена архитектура биллинговой системы. В этом разделе подробно рассмотрена общая архитектура системы, интерфейсы компонентов и детали их реализации.

Четвертая глава посвящена реализации биллиногово й системы с использованием технологий J2EE и Oracle ADF.

В пятой главе приводятся результаты тестирования биллиногово й системы.

В заключении сделаны выводы о проделанной работе.

Приложение содержит детальное описание прецедентов для каждого пользователя системы.

# Обзор технологий посторения веб-приложений

## Язык PHP

UML (Unified Modeling Language) [10] – это универсальный язык визуального моделирования программных систем. UML предназначен для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем.

Унифицированный процесс (Unified Process)– методология моделирования программных систем. UP определяет 5 основных рабочих потоков: определение требований, анализ, проектирование, реализация и тестирование, которые представляют собой одну итерацию [2]. UP имеет 4 фазы, каждая из которых может состоять из одной или нескольких итераций.

Нафазе *«начало»* команда разработчиков, совместно с заказчиком и потенциальными пользователями системы определяют цели разработки программного обеспечения. Основная задача данной фазы — определить требования к разрабатываемой системе. Требования к системе могут быть описаны при помощи *UML-диаграммы прецедентов* (вариантов использования) [4] и уточнены посредством развернутых прецедентов.

*Прецедент (вариант использования)* представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом [2]. Прецеденты описывают типичные варианты взаимодействия между актерами и системой.

*Актер -*  множество логически связанных ролей в UML, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс).

*Развернутые прецеденты* (expanded use case) [4] представляют собой более подробное детальное описание того, что должна делать система. Развернутые прецеденты описываются в виде таблицы и содержат информацию о кратком содержании прецедента, актерах, фактических этапах и состоянии системы до и после выполнения прецедента.

В *фазе «Уточнение»* производится анализ предметной области и построение исполняемой архитектуры при помощи UML-диаграмм классов, последовательностей, развертывания.

*Диаграммы классов* являются одной из форм статического описания системы с точки зрения ее проектирования, показывая ее структуру. На диаграммах классов показываются классы, интерфейсы и отношения между ними.

*Диаграммы последовательности* используются для моделирования взаимодействия объектов во времени. На диаграмме последовательности изображаются только те объекты, которые непосредственно участвуют во взаимодействии. Ключевым моментом для диаграмм последовательности является динамика взаимодействия объектов во времени.

*Диаграмма развертывания* предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками.

В *фазе «Построение»* происходит реализация большей части функциональности продукта.

В *фазе «Внедрение»* создается финальная версия продукта и передается от разработчика к заказчику. Это включает в себя программу бета-тестирования, обучение пользователей, а также определение качества продукта.

На сегодняшний день PHP лидирует, как язык веб-программирования – на нем работает 81,2 % сайтов (по которым удалось получить информацию о серверной стороне, по данным w3techs.com/technologies/overview/programming\_language/all), также PHP входит в пятерку лидеров по общему рейтингу языков программирования (по данным www.tiobe.com/index.php/content/paperinfo/tpci/index.html). На PHP работают yahoo.com, wikipedia.org, flickr.com, Friendster.com, source-forge.net, amazon.com, istockphoto.com, facebook.com, twitter.com, vk.com, mail.ru. PHP поддерживается большинством хостинг-провайдеров, взаимодействует с большим количеством СУБД (MySQL, SQLite, PostgreSQL, и др.), и является довольно простым в освоении. История PHP началась в 1994 году, когда датский программист Расмус Лерфорд создал набор скриптов на Perl/CGI для вывода и учета посетителей его онлайн резюме, который Лер-форд назвал Personal Home Page. Вскоре разработчику стало тесно в грани-цах Perl, вследствие чего был разработан новый интерпретатор шаблонов PHP/FI (Personal Home Page/Form Interpreter). В 1997 году PHP/FI 2.0 ис-пользовали около 1% всех интернет-доменов мира (приблизительно 50 ты-сяч). С тех пор язык заметно преобразился и претерпел множество версий и изменений. Актуальная версия на сегодняшний день - PHP 5.5.

Чтобы приложение на PHP работало, необходим веб-сервер. Разработка будет вестись сразу с хостинга, без использования локального сервера, что позволяет не задумываться о миграции, при разработке сайт будет привязан к тестовому домену, когда же работа будет завершена, он прилинкуется к рабочему имени. На хостинге используется сервер Apache, есть поддержка MySQL 5, PHP 5. Так же по умолчанию установлено phpMyAdmin – приложение для администрирования баз данных на MySQL. Чаще всего веб-приложение состоит из трех компонент:

1. Клиентская часть веб-приложения
2. Серверная часть веб-приложения
3. База данных

По запросу из клиентской части серверная часть вынимает из базы данных нужную информацию и предоставляет ее конечному пользователю.

## Content Management Frameworks (CMF)

Выбор системы управления баз данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений, обеспечивающих работу со структурированными данными. Выбранный программный продукт должен удовлетворять как текущим, так и будущим потребностям предприятия, при этом следует учитывать финансовые затраты на приобретение необходимого оборудования, самой системы, разработку необходимого программного обеспечения на ее основе, а также обучение персонала [7].

На сегодняшний день существует большое количество вариаций систем управления базами данных, различающихся архитектурой, размером, применимостью, количеством пользователей и возможностью одновременной работы с БД. Современные реляционные базы данных можно разделить на три группы:

1. персональные – реляционные базы данных, основанные на архитектуре файл-сервер, имеющие небольшое количество пользователей (MS Access [13], Paradox, dBase, FoxPro);
2. корпоративные – реляционные базы данных, основанные на архитектуре SQL – сервер и применяющиеся в крупных компаниях с достаточно большим количеством пользователей [5] (Oracle [1], MS SQL Server [12], DB2, MySQL [14], Postgre SQL);
3. параллельные – реляционные базы данных, основанные на кластерных системах, больших вычислительных комплексах (DB2 Parallel Edition, Teradata, NonStop SQL, VoltDB).

Для реализации веб-приложения, работающего с биллинговой системой энергосбытовой компании была выбрана база данных Oracle Database 11g [5]. Большое количество крупных корпораций работают именно с СУБД компании Oracle, поскольку это быстрая, надежная, безопасная и легкая в управлении система.

Для упрощения разработки веб-приложений на PHP существуют так называемые Content Management Frameworks (CMF), которые за последнее время набрали популярность и стали базовой платформой для разработки веб-приложений. CMF – это своего рода каркас для построения веб-приложений. Использование PHP-фреймворков позволяет экономить большое количество времени, уменьшить нагрузку на процесс разработки, повторно использовать код. В отличие от Content Management Systems (CMS), CMF позволяет создать более безопасный, гибкий, простой в использовании сайт. CMF, как и CMS представляет собой набор модулей, но в фрэймворке модули можно настраивать более детально, нежели в CMS, более того, неиспользуемые модули можно даже полностью удалить из системы, в то время как в CMS это не всегда возможно. При этом сложности разработки на CMF вполне сравнима со сложностью создания сайта на CMS. Таким образом CMF является более гибким инструментом для создания веб-приложений.

## MVC

Клиент-сервер — архитектура вычислительной системы, в которой задачи или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами [6] . Клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и [программным обеспечением](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Классическая (двухзвенная) клиент-серверная архитектура представляет собой информационную систему, основанную на использовании серверов баз данных. Обычно сервер в этом случае является лишь [хранилищем данных](http://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), а вся работа по обработке и представлению этих данных переносится на машину клиента. При использовании такого подхода возникают некоторые сложности при установке, настройки и обслуживании клиентского приложения, вследствие чего невозможно обеспечить масштабируемость приложения, которая так важна при разработке сложных корпоративных приложений [1].

Многоуровневая клиент-серверная архитектура – это разновидность клиент-серверной архитектуры, в которой функции обработки данных вынесены на один или несколько серверов. Среди многоуровневой архитектуры наиболее распространена трехзвенная архитектура, предполагающая наличие следующих компонентов приложения: клиентское приложение, сервер приложений и сервер базы данных. Этот подход является гораздо эффективнее двухзвенной архитектуры и обладает рядом преимуществ [6]:

1. высокая безопасность и надежность приложений;
2. конфигурируемость – изолированность уровней друг от друга позволяет быстро переконфигурировать систему при возникновении сбоев;
3. возможность не устанавливать клиентское программное обеспечение на все компьютеры, которые работают с приложением, что существенно снижает затраты на развертывание такой системы;
4. администрирование одного сервера приложений намного проще, чем множества клиентских компьютеров;
5. достигается высокая масштабируемость приложений за счет удобства администрирования.

Большинство современных CMF являются реализацией архитектуры Model-View-Controller (MVC), которая позволяет сконцентрироваться на реализации бизнес-логиги, меньше уделяя внимания собственно программированию. Концепция MVC была описана задолго до появления PHP, в 1979 году, программистом Трюгве Реенскауг (англ. Trygve Reenskaug). Основная цель применения MVC состоит как раз в том, чтобы отделить бизнес-логику (модель) от её визуализации (представления/вида). При этом контроллер обеспечивает связь между пользователем и системой: отслеживает действия пользователя и реализует необходимую реакцию, посредством модели и представления. Представление и контроллер зависят от модели, однако модель не зависит ни от того, ни от другого. Это позволяет строить модель независимо от ее визуального представления, а так же создавать несколько представлений для одной модели.

Среди php-фрейворков, работающих по данной модели наиболее популярны: Yii, CodeIgniter, Zend, CakePHP, Symfony. В данной работе я буду использовать CodeIgniter 2.1.4 (CI), т.к. этот фреймворк имеет ряд преимуществ:

1. Достаточно развитые англо и русскоязычные сообщество;
2. Отличная документация и множество примеров;
3. Простота установки и легкость в освоении;
4. Гибкость;
5. Жесткая структура каталогов;
6. CI не слишком громоздкий, в то же время – это не микрофрэймворк, т.е. на нем можно строить достаточно большие приложения;
7. Хорошие результаты на benchmark-тестах (<http://systemsarchitect.net/performance-benchmark-of-popular-php-frameworks/>).

## Варианты построения информационных систем с веб-интерфейсом

Для создания веб-интерфейсов можно воспользоваться средствами быстрой разработки приложений, такими как Oracle Application Express (APEX) [11]. В APEX применяется принцип декларативного программирования, при котором код не генерируется, не компилируется, а пользователь взаимодействует с мастерами и списками свойств [9]. Однако существенным минусом таких интерфейсов является их шаблонность.

Другой способ разработки веб-клиента основан на традиционном программировании. Веб-приложения, работающее с биллинговой системой энергосбытовой компании, будет написано на языке Java в среде JDeveloper [1] с использованием технологий JavaEE [1] и ADF [1].

Oracle JDeveloper – интегрированная среда разработки для моделирования, разработки и отладки J2EE-приложений и веб-сервисов [5]. Разработка приложений в JDeveloper существенно упрощена за счет использования мастеров, редакторов, инструментов моделирования. Также эта среда поддерживает возможность интерактивной привязки данных к пользовательскому интерфейсу и автоматического развертывания в сервере приложений.

Oracle JDeveloper имеет набор компонентов для разработки веб-приложений - Application Development Framework(ADF). Jdeveloper хорошо интергрирован с ADF, а также с сервером приложений OC4J, что позволяет создавать сложные корпоративные приложения более быстро и эффективнее.

J2EE (Java 2 Platform, Enterprise Edition) – набор стандартов и спецификаций (платформа), созданная и поддерживаемая компанией Sun Microsystem. J2EE описывает компоненты программных приложений, предназначенных для целой организации. J2EE имеет дополнительные функциональные возможности, касающиеся развертывания многозвенной модели и web-технологий. Поэтому J2EE пользуется популярностью в тех организациях, где необходимо разрабатывать приложения, которые будут работать в Интернете [3].

## Вывод

Изучив существующие способы построения информационных систем, проанализировав их главные достоинства и недостатки, а также в соответствии с требованиями заказчика, для реализации приложения, работающего с биллинговой системой энергосбытовой компании, было решено разрабатывать веб-приложение в среде Oracle JDeveloper 10g с использованием технологий Oracle ADF и J2EE.

# Требования к веб-приложению интернет-магазина

## Функциональные требования

Компания занимается продажей сувенирной продукции. В рамках данной системы единственным контрагентом компании является Клиент.

Функциональные требования определяют функциональность программного обеспечения, то есть описывают, какое поведение должна предоставлять разрабатываемая система. Функциональные требования включают в себя бизнес-требования и пользовательские требования. В ходе интервью с заказчиком были выявлены следующие функциональные требования для реализации веб-приложения:

1. Обеспечивать свободный доступ и поиск информации о компании и ее продукции для потенциальных клиентов.
2. Обеспечивать возможность самостоятельного формирования заказа клиентом.
3. Хранить информацию о клиентах и их заказах.
4. Обеспечивать обратную связь клиента и менеджера компании.
5. Обеспечивать возможность добавления, удаления, редактирования информации на сайте менеджером компании.

## Нефункциональные требования

Нефункциональные требования описывают свойства и ограничения, накладываемые на информационную систему. Нефункциональные требования определяют бизнес-правила, системные требования и т.д. Для реализации веб-приложения интернет-магазина были зафиксированы следующие нефункциональные требования:

1. Система должна быть разработана с использованием языка PHP.
2. Система должна быть реализована в соответствии с MVC технологией построения веб-приложений.

## Прецеденты использования системы

Попадая на сайт, клиент может без регистрации начать просмотр его разделов:

1. Главная страница (Вывод произвольных товаров)
2. Новости магазина
3. Обратная связь
4. Корзина

Клиент может добавлять товары в корзину, удалять их из нее, в пределах еще не оформленного заказа. Клиент может оставить сообщение через форму обратной связи. Клиент может зарегистрироваться, а так же войти, как уже зарегистрированный пользователь. При оформлении заказа система проверяет, авторизовался ли клиент в системе, если да, то заказ оформляется, если нет, то система предлагает зарегистрироваться.

Так же с сайтом может работать менеджер. Чтобы были доступны дополнительные возможности менеджера нужно авторизоваться, как работник компании. Добавить менеджера на сайт может глобальный администратор. Менеджер может просматривать каталог с категориями товаров и редактировать его, просматривать каталог товаров и редактировать его, просматривать и редактировать отдельно взятые товары. Менеджер может видеть список заказов и менять их статус. Менеджер может просматривать список клиентов сайта. Так же менеджер управляет блогом сайта.Можно выделить шесть основных актеров, взаимодействующих с разрабатываемой биллинговой системой (Рис. 1):

*Работник договорного отдела* регистрирует и обновляет информацию о контрагентах и заключенных с ними договорах купли-продажи услуг компании.

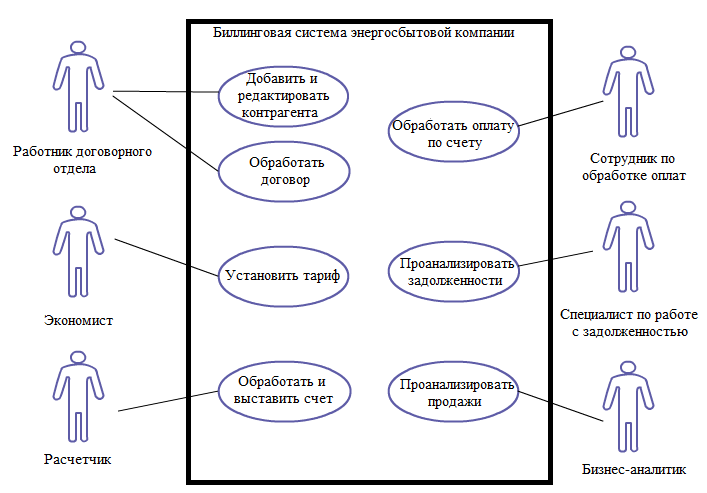


Рис. 1. Диаграмма прецедентов биллинговой системы   
энергосбытовой компании

*Экономист* устанавливает новые тарифы или редактирует существующие цены на услуги компании.

*Расчетчик* обрабатывает данные, считанные с показаний приборов учета, и на основе этих данных выставляет счета по договорам и отправляет эти счета контрагентам.

*Сотрудник по обработке оплат* обрабатывает поступившие оплаты с целью подсчета задолженности по каждому договору.

*Специалист по работе с задолженностью* анализирует информацию о должниках энергосбытовой компании.

*Бизнес-аналитик* оценивает работу компании на основе отчетов по продажам услуг энергосбытовой компании.

С каждым актером системы связан определенный прецедент. На данной диаграмме некоторые низкоуровневые прецеденты объединены в прецеденты более высокого порядка.

*Добавить и редактировать контрагента* – добавить информацию о новом клиенте компании, являющимся физическим или юридическим лицом, в базу данных или изменить существующие данные о клиенте.

*Обработать договор* – добавить данные по новому заключенному договору энергосбережения или изменить данные уже существующего в базе данных договора.

*Установить тариф* – изменить данные по тарифам, на основе которых выполняются авансовые и итоговые начисления клиентам компании.

*Обработать и выставить счет* – выполнить авансовые и итоговые начисления клиентам энергосбытовой компании.

*Обработать оплату по счету* - произвести операцию разноски оплаты, т.е. связать поступившую оплату и счета потребителя.

*Проанализировать задолженности* – сформировать отчет о клиентах, имеющих наибольшую задолженность на основе информации об оплатах.

*Проанализировать продажи* – сформировать отчет о продажах энергии компании поквартально, за месяц, за год.

Более подробное описание прецедентов для каждого актера, взаимодействующего с биллинговой системой, представлено в Приложении 1.

## Вывод

В процессе беседы с заказчиком было составлено техническое задание, в котором сформулированы основные функциональные и нефункциональные требования, предъявляемые к разрабатываемой системе, определены главные пользователи системы и варианты использования системы пользователями.

# Архитектура веб-приложения биллинговой системы

## Общий интерфейс системы

Интерфейс веб-приложения предоставляет отдельные методы работы с системой в зависимости от того, какими правами обладает пользователь, вошедший в систему:

* интерфейс работника договорного отдела;
* интерфейс работника-экономиста;
* интерфейс расчетчика;
* интерфейс сотрудника по обработке оплат
* интерфейс специалиста по работе с задолженностью
* интерфейс бизнес-аналитика

Один пользователь может получить доступ сразу к нескольким интерфейсам, если соответствующие роли прописаны в его пользовательском профиле.

Диаграмма доступных пользовательских интерфейсов представлена на Рис. 2.

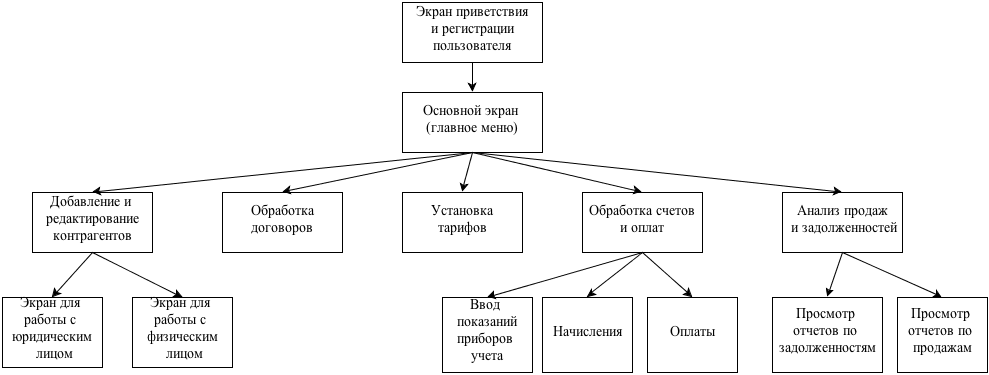


Рис. 2. Диаграмма доступных пользовательских интерфейсов

## Интерфейс для работника договорного отдела

Работник договорного отдела заключает новые договора с контрагентами или расторгает существующие. Соответственно он должен заполнять (обновлять, удалять) информацию о контрагентах, с которыми сотрудничает компания, и обрабатывать данные о договорах, которые заключены с контрагентами (Рис. 3).

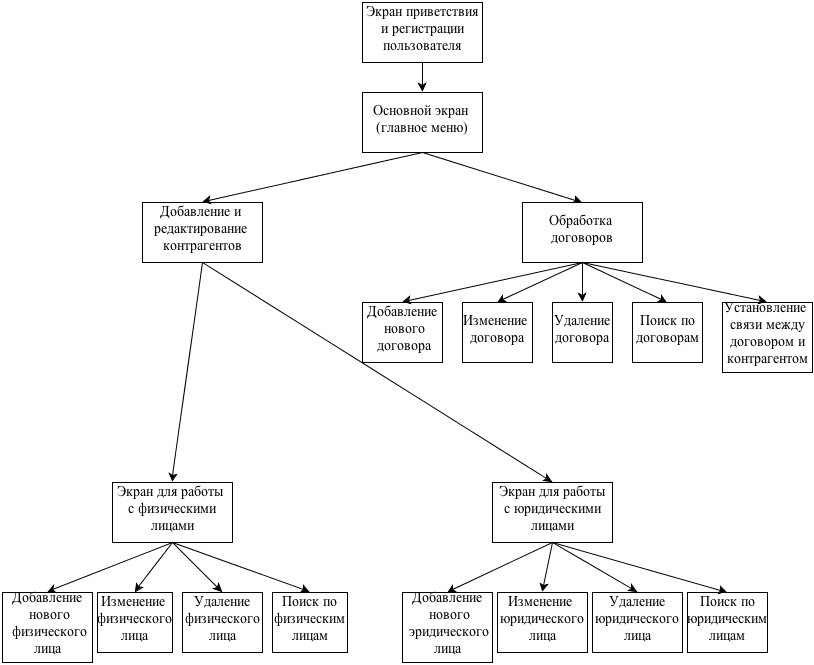


Рис. 3. Возможности работы с системой для работника договорного отдела

## Интерфейс для экономиста

Экономист компании устанавливает тарифы по оплате услуг, предоставляемых компанией. Наряду с установлением цен на новые услуги, экономист может изменить или удалить существующие тарифы (Рис. 4).

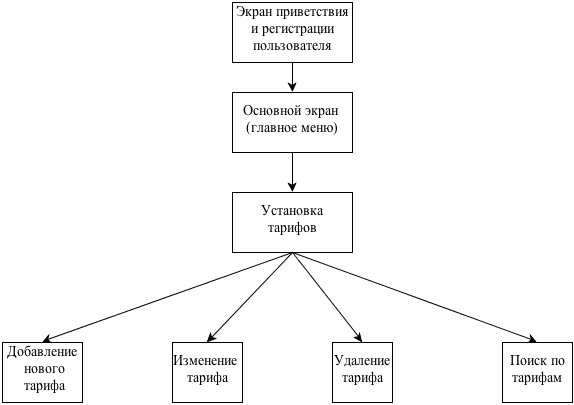


Рис. 4. Возможности работы с системой для экономиста

## Интерфейс для расчетчика

Расчетчик энергосбытовой компании снимает показания с приборов учета (счетчиков) и вводит эти данные в базу. Расчетчик также имеет права на редактирование или удаление устаревших данных.

Второй задачей расчетчика является выполнение начислений (авансовых и итоговых) в соответствии с показаниями приборов учета и тарифами на предоставление услуг компании, которые устанавливает экономист. По результатам полученных данных расчетчик может зафиксировать операцию, т.е. выставить счет контрагенту, или отменить полученные счета, если они заведомо ложные (Рис. 5).

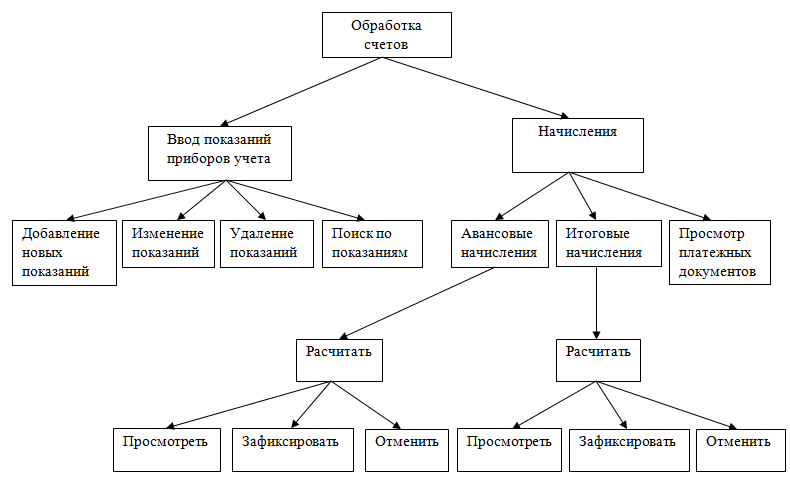


Рис. 5. Возможности работы с системой для расчетчика

## Интерфейс для сотрудника по обработке оплат

Сотрудник по обработке оплат ведет учет поступивших оплат на счет энергосбытовой компании. Он может добавлять новые оплаты вручную или осуществлять импорт оплат посредством специализированно программы.

Сотрудник по обработке оплат производит операцию разноски оплаты, т.е. в первую очередь, соотносит оплату с конкретным договором (один контрагент может заключить несколько договоров), а во вторых связывает оплату с конкретным начислением внутри договора с тем, чтобы в будущем могла быть подсчитана задолженность по каждому договору (Рис. 6).

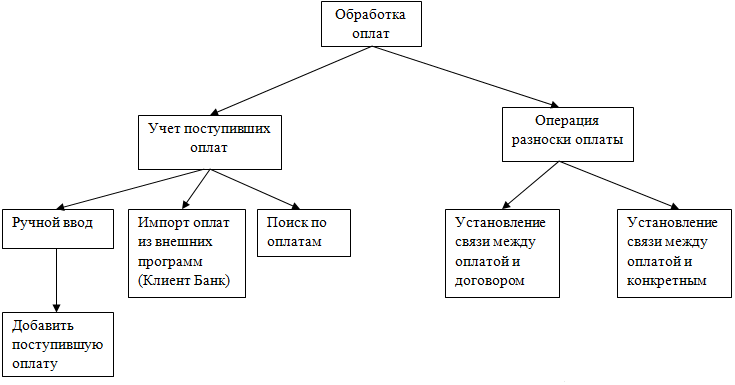


Рис. 6. Возможности работы с системой для сотрудника по обработке оплат

## Интерфейс для специалиста по работе с задолженностью

Специалист по работе с задолженностью может только просматривать отчеты по конкретным должникам, а также по структуре, динамике задолженности в целом. Эти отчеты очень важны для компании, поскольку по результатам отчетов сотрудник должен выработать эффективную политику по работе с должниками. Иначе энергосбытовая компания рискует остаться банкротом (Рис. 7).

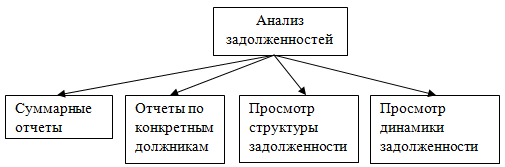


Рис. 7. Возможности работы с системой для специалиста по работе с задолженностью

## Бизнес-аналитик

Бизнес-аналитик может лишь просматривает отчеты по продажам услуг энергосбытовой компании. По этим отчетам делается вывод о работе компании и вырабатывается дальнейшая бизнес-стратегия (Рис. 8).

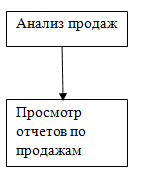


Рис. 8. Возможности работы с системой для бизнес-аналитика

## Вывод

В процессе детального изучения предметной области были спроектированы интерфейсы для работы с системой возможных пользователей. Интерфейсы включают в себя подробное описание возможностей работы с системой каждого пользователя.

# Реализация веб-приложения биллинговой системы

## Схема базы данных

Для реализации веб-приложения для работы с биллинговой системой энергосбытовой компании была использована база данных, состоящая из следующих таблиц:

* таблица контрагентов, содержащая общую информацию обо всех потребителях энергосбытовой компании;
* таблица физических лиц, содержащая информацию о клиентах-людях;
* таблица юридических лиц, содержащая информацию о клиентах-компаниях;
* таблица сторон договора, связывающая каждого контрагента с конкретным договором;
* таблица договоров, содержащая информацию о заключенных договорах;
* таблицы авансовых и итоговых начислений, связывающие каждое начисление с конкретным договором;
* таблица оплат, связывающая поступившие оплаты с конкретным договором;
* таблица товаров, содержащая информацию о товарах и услугах, предоставляемых энергосбытовой компанией;
* таблица тарифов, содержащая информацию о ценах на услуги компании.

Схема используемой базы данных изображена на Рис. 9.

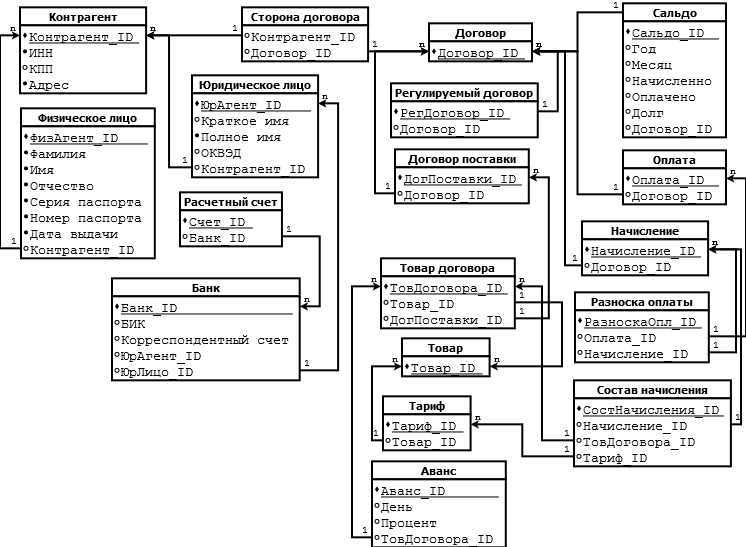


Рис. 9. Схема базы данных

## Файловая схема проекта

Проект реализован на основе шаблона Модель-Представление-Контроллер (Model-View-Controller, MVC), который определяет разделение кода приложения на 3 слоя:

1. Модель (Model\Application Sources). Слой модели определяет и проверяет достоверность данных, используемых приложением. В него входят программы, проверяющие бизнес-правила и поддерживающие связь с базой данных для передачи информации. Этот слой взаимодействует со слоем представления, сообщая об изменении данных. Он также работает со слоем контроллера, принимая и обрабатывая запросы на обновление данных в приложении.
2. Представление(ViewController\Web Content). Слой представления поддерживает интерфейс пользователя, обеспечивая отображение данных, полученных от слоя модели. Он взаимодействует со слоем контроллера, принимая запросы на конкретные отображения (страницы) и посылая ему сообщения о событиях, связанных с действиями пользователя.
3. Контроллер(ViewController\Application Sources). Программа слоя контроллера определяет, что будет происходить в результате действия пользователя на слое представления. Слой контроллера взаимодействует со слоем представления, запрашивая страницы, и отправляет запросы слою модели, когда данные необходимо обновить.

Файловая система веб приложения для организации работы биллинговой системы энергосбытовой компнаии представлена на Рис. 10.

В узле Application Sources/model.view.entities представлены бизнес-сущности для работы с объектами базы данных.

В узле Application Sources/ebyt.view.framework находится java-файлы, содержащие код для обработки данных и выполнения операция с ресурсами. Например, файл Logout.java определяет реакцию на событие при нажатии глобальной кнопки logout.

В узле Web Content располагаются существующие веб-страницы приложения. В папках pages, help находятся защищенные страницы, доступные всем авторизированным пользователям. В папке security лежат особые страницы login.jsp и error.jsp, отвечающие за безопасность приложения. Защищенные страницы сотрудников энергосбытовой компании разделены на разные физические директории и помещены в папки pages\admin, pages\analist, pages\contracter, pages\counter, pages\debts, pages\economist. В папке image лежат картинки, используемые при создании jsp-страниц веб-приложения.

В узле WEB-INF находятся 2 важнейших файла: faces-config.xml – конфигурационный файл приложения, который позволяет определять правила перехода между страницами и автоматически генерируемый файл web.xml, код которого можно изменять главным образом для настройки системы безопасности.

В папках WEB-INF\lib и WEB-INF\temp\adf расположены используемые библиотеки и шаблоны для создания приложения.

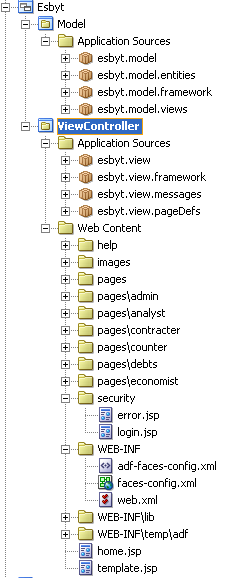


Рис. 10 Файловая система проекта

## Организация безопасности приложения

Безопасность приложения осуществляется на основе создания пользователей и ролей. Ролям соответствуют логические роли внутри приложения. Логические роли описывают правила, согласно которым разрешается доступ к различным частям приложения. Каждому пользователю назначается одна или несколько логических ролей.

Подключение к приложению происходит в два этапа:

1. Аутентификация. Система безопасности проверяет права доступа пользователя на основании его имени и пароля.
2. Авторизация. После прохождения этапа авторизации система безопасности передает приложению информацию о пользователе в виде списка предоставленных ему ролей.

В зависимости от имени подключившегося пользователя и его прав можно скрыть или сделать недоступными определенные части приложения.

Для того чтобы иметь возможность подключится к системе, каждый пользователь должен быть зарегистрирован в файле jazn-data.xml. Пример кода файла jazn-data.xml представлен на Рис. 11.

<users>

<user>

<name>UCounter</name>

<credentials>{903}aCKi...cTHY3fCQI=</credentials>

</user>

<user>

<name>UEconomist</name>

<credentials>{903}Lqe...upIXAM=</credentials>

</user>

<user>

<name>UAnalyst</name>

<credentials>{903}0LlK...raM</credentials>

</user>

<user>

<name>UAdmin</name>

<credentials>{903}GOij...LNyPD1</credentials>

</user>

<user>

<name>UDirector</name>

<credentials>{903}xVy...QTX1s=</credentials>

</user>

<user>

<name>UContracter</name>

<credentials>{903}CPnZ...XpptA=</credentials>

</user>

</users>

Рис. 11 Создание пользователей и паролей

После того, как созданы пользователи, нужно создать роли и распределить по ним пользователей (Рис. 12). В веб-приложении для организации работы биллинговой системы создано шесть ролей:

1. admin – роль администратора приложения;
2. contracter – роль для работников договорного отдела;
3. economist – роль для экономистов компании;
4. counter – роль для расчетчиков;
5. specialist from debts – роль для специалистов по работе с задолженностями;
6. analyst – роль для бизнес-аналитиков.

<role>

<name>admin</name>

<members>

<member>

<type>user</type>

<name>UAdmin</name>

</member>

</members>

</role>

<role>

<name>counter</name>

<members>

<member>

<type>user</type>

<name>UCounter</name>

</member>

<member>

<type>user</type>

<name>UDirector</name>

</member>

</members>

</role>

Рис. 12 Пример назначения ролей пользователям

Модель безопасности приложения разделяет все страницы на несколько подмножеств в зависимости от того, какие группы пользователей могут иметь к ним доступ.

1. *Не подключившиеся пользователи.* Этим пользователям доступна только начальная страница.
2. *Сотрудники договорного отдела.* Этим пользователям доступны начальная страница в состоянии после входа в систему и страницы просмотра и редактирования данных по контрагентам и договорам (по закладке Contracter).
3. *Экономисты.* Этим пользователям доступны начальная страница в состоянии после входа в систему и страницы просмотра и редактирования тарифов (по закладке Economist).
4. *Расчетчики.* Этим пользователям доступны начальная страница в состоянии после входа в систему и страницы просмотра и редактирования данных по показаниям приборов учета и начислений (по закладке Counter).
5. *Специалисты по работе с задолженностями.* Этим пользователям доступны начальная страница в состоянии после входа в систему и страницы просмотра отчетов по текущим задолженностям (по закладке Specialist from Debts).
6. *Бизнес-аналитики.* Этим пользователям доступны начальная страница в состоянии после входа в систему и страницы просмотра отчетов по продажам компании (по закладке Anlyst).
7. *Руководители.* Этим пользователям доступны начальная страница в состоянии после входа в систему и все страницы работников энергосбытовой компании.
8. *Администраторы.* Этим пользователям доступны все страницы приложения, включая страницы справочников и должностей (по закладке Admin).

Задача ограничения доступа к страницам решается путем разделения несколько функциональных областей приложения на несколько физических директорий относительно корневого каталога веб-приложения.

# Тестирование веб-приложения биллинговой системы

## Тестирование системы безопасности приложения

Первый этап тестирования приложения начинается с проверки правильности аутентификации и авторизации пользователей. Неавторизованные пользователи могут просматривать только незащищенную начальную страницу, представленную на Рис. 13.

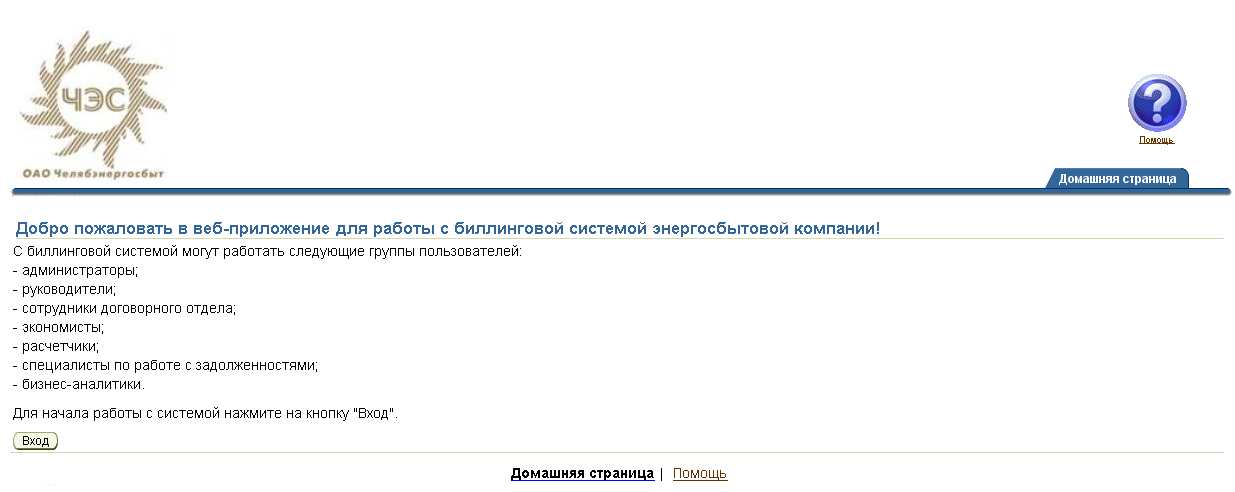


Рис. 13 Незащищенная начальная страница

При попытке пользователя зайти в систему выскакивает форма авторизации пользователей и запрашивает у пользователя имя и пароль (Рис. 14).

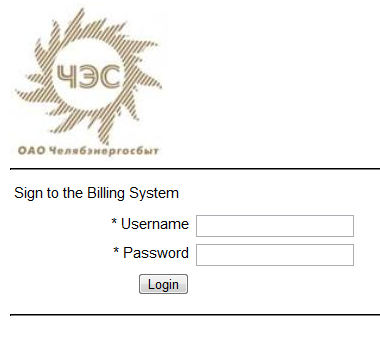


Рис. 14 Форма авторизации пользователей

В случае неправильного значения полей имени пользователя или пароля появляется сообщение о неверности данных (Рис. 15).



Рис. 15 Ошибка авторизации

В случае успешной авторизации пользователя происходит переход на защищенную домашнюю страницу (Рис. 16).

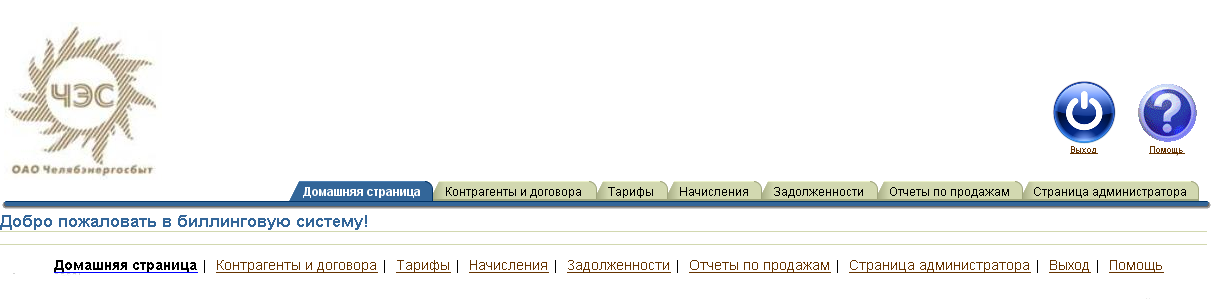


Рис. 16 Защищенная домашняя страница

На защищенной домашней странице посредством вкладок можно выбрать страницу для работы сотрудника энергосбытовой компании. Если у пользователя достаточно прав, чтобы просматривать выбранную страницу, то переход на страницу произойдет (Рис. 17). В противном случае пользователя снова перебросит на форму авторизации.



Рис. 17 Страница для работы с данными для сотрудника договорного отдела

Для выхода из системы следует нажать глобальную кнопку «Выход». Появится диалог с просьбой подтвердить желание выйти из приложения (Рис. 18). При нажатии на кнопку «да» сессия пользователя будет прервана и снова появится незащищенная домашняя страница.

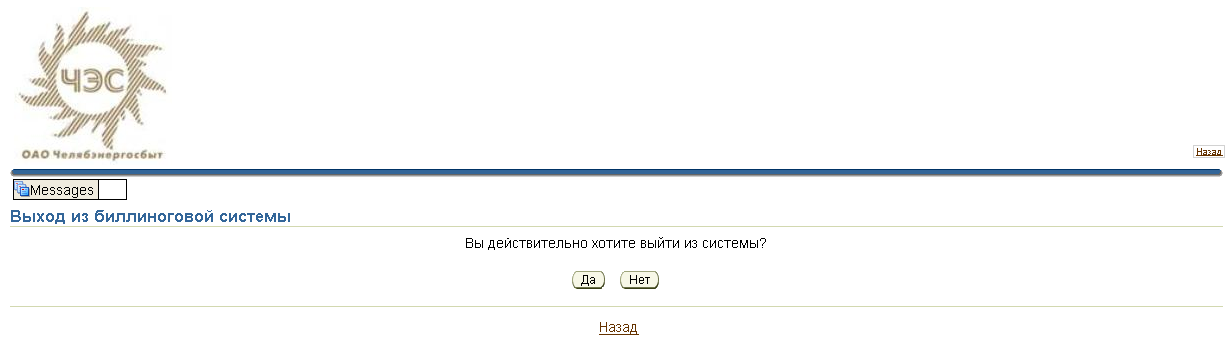


Рис. 18 Страница выхода из системы

## Тестирование пользовательского интерфейса системы

Второй этап тестирования заключается в проверки правильности работы системы со стороны каждого пользователя системы. После успешной авторизации пользователя он получает доступ к своей рабочей странице (Рис. 19).

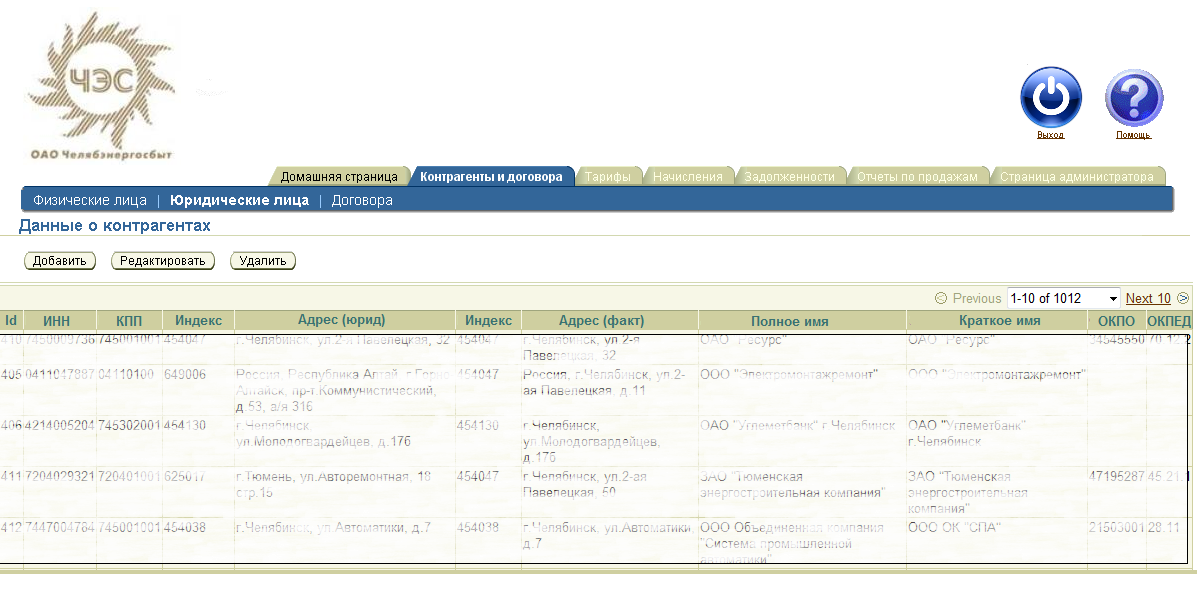


Рис. 19 Интерфейс сотрудника договорного отдела

Авторизованный пользователь (сотрудник договорного отдела) может переходить по всем вкладкам на своей рабочей странице, добавлять, изменять и удалять имеющиеся данные посредством кнопок «Добавить», «Редактировать» и «Удалить»(Рис. 20 и Рис. 21).

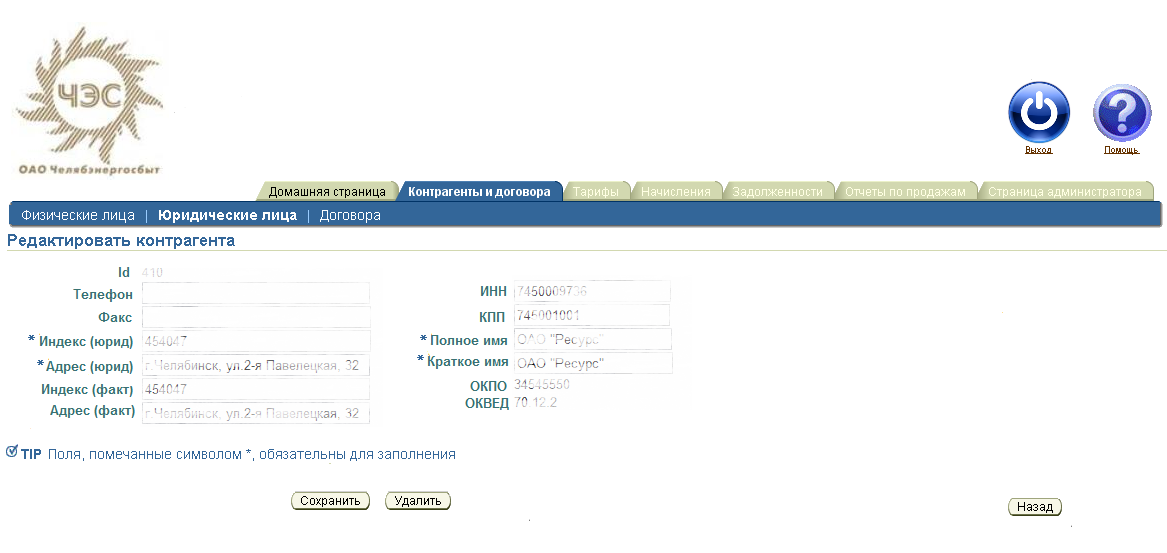


Рис. 20 Редактирование контрагента



Рис. 21 Редактирование договора

# Заключение

В ходе проделанной работы мной были изучены существующие подходы, применяющиеся при построении информационных систем, выбран наиболее оптимальный из них и на его основе реализовано веб-приложение для организации работы биллинговой системы энергосбытовой компании. В процессе работы были изучены возможности работы со средой программрования Oracle JDeveloper и технологии программирования J2EE и Oracle ADF.

Было произведено тестирование приложения на реальных пользовательских данных. Приложение успешно прошло предложенные тесты.

Разработанное веб-приложение при небольших доработках в дальнейшем можно внедрять в эксплуатацию энергосбытовой компании взамен существующего устаревшего приложения, основанное на двухзвенной архитектуре клиент-сервер.

# Литература

1. Колетцки П., Миллс Д. Oracle JDeveloper 10g Руководство по разработке Интернет-приложений J2EE с помощью Oracle JDeveloper и Oracle ADF. – М.: Лори, 2012. –574 с.
2. Арлоу Д., Нейштадт А. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. 2-е издание – СПб: Символ-плюс, 2007. –624 с.
3. Сьерра К., Бейтс Б. Изучаем Java. 2-е издание. – М.: Эксмо, 2012. –708 с.
4. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. –352 с.
5. Кайт Т. Oracle для профессионалов: Архитектура, методики программирования и особенности версий 9i, 10g, 11g. – М.: Вильямс, 2011. –848 с.
6. Кайт Т. Эффективное проектирование приложений Oracle. – М.: Лори, 2008. –656 с.
7. Гарсиа–Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс. – М.: Вильямс, 2003. – 1088 с.
8. Бенкен Е. PHP, MySQL, XML. Программирование для интернета. 3-е издание. – СПб: БХВ–Петербург, 2011. – 304 с.
9. Бобровский С. Oracle Database XE для Windows. Эффективное использование. – М.: Лори, 2009. – 512 с.
10. Буч Г., Рамбо Д., Ябсон И. Язык UML. Руководство пользователя. 2-е издание – М.: ДМК Пресс. -496 с.
11. Rick Greenwald. Beginning Oracle Application Express. Wrox, 2008. -384 с.
12. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих. СПб.: БХВ-Петербург, 2013. -816 с.
13. Бекаревич Ю. Пушкина Н. Aceess 2007. - СПб.: БХВ-Петербург, 2007. -720 с.
14. Дюбуа П. My SQL. 3-е издание. –М.: Вильямс, 2007. -1168 с.

# Приложение 1. Описание прецедентов

**Таб. 1** Описание прецедента AddPerson

|  |  |
| --- | --- |
| Имя прецедента | Прецедент AddPerson |
| Идентификатор прецедента | ID: 1 |
| Краткое описание | Добавить новое физическое лицо |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Заключение договора с новым физическим лицом, получение данных о новом физическом лице |
| Фактические этапы прецедента | 1. Проверить наличие физического лица в системе (по ИНН)  2. Ввести данные по новому физическому лицу (ФИО, адрес, серия и номер паспорта, ИНН и другие реквизиты) |
| Состояние системы после окончания прецедента | Информация о новом физическом лице введена в базу данных |
| Альтернативные потоки | Выдать сообщение о наличие в системе физического лица с таким же ИИН |

**Таб. 2** Описание прецедента EditPerson

|  |  |
| --- | --- |
| Имя прецедента | Прецедент EditPerson |
| Идентификатор прецедента | ID: 2 |
| Краткое описание | Изменить данные о существующем в базе данных физическом лице |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Получены новые данные по существующему в базе данных физическому лицу |
| Фактические этапы прецедента | 1. Ввести измененные данные физического лица |
| Состояние системы после окончания прецедента | Информация о физическом лице обновлена |
| Альтернативные потоки | Нет |

**Таб. 3** Описание прецедента AddCompany

|  |  |
| --- | --- |
| Имя прецедента | Прецедент AddCompany |
| Идентификатор прецедента | ID: 3 |
| Краткое описание | Добавить новое юридическое лицо |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Заключение договора с новым юридическим лицом, получение данных о новом юридическом лице |
| Фактические этапы прецедента | 1. Проверить наличие юридического лица в системе (по КПП)  2. Ввести данные по новому юридическому лицу (название, адрес, КПП и другие реквизиты) |
| Состояние системы после окончания прецедента | Информация о новом юридическом лице введена в базу данных |
| Альтернативные потоки | Выдать сообщение о наличие в системе юридического лица с таким же КПП |

**Таб. 4** Описание прецедента EditCompany

|  |  |
| --- | --- |
| Имя прецедента | Прецедент EditCompany |
| Идентификатор прецедента | ID: 4 |
| Краткое описание | Изменить данные о существующем в базе данных юридическом лице |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Получены новые данные по существующему в базе данных юридическому лицу |
| Фактические этапы прецедента | 1. Ввести измененные данные юридического лица |
| Состояние системы после окончания прецедента | Информация о юридическом лице обновлена |
| Альтернативные потоки | Нет |

**Таб. 5** Описание прецедента AddContract

|  |  |
| --- | --- |
| Имя прецедента | Прецедент AddContract |
| Идентификатор прецедента | ID: 5 |
| Краткое описание | Добавить новый договор энергоснабжения |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Заключен новый договор энергоснабжения |
| Фактические этапы прецедента | 1. Проверить наличие данного договора в системе (по номеру договора)  2. Ввести данные по новому заключенному договору (номер, вид, даты заключения и тд) |
| Состояние системы после окончания прецедента | Информация о новом договоре введена в базу данных |
| Альтернативные потоки | Выдать соответствующее сообщение о наличие договора в системе с таким же номером |

**Таб. 6** Описание прецедента EditContract

|  |  |
| --- | --- |
| Имя прецедента | Прецедент EditContract |
| Идентификатор прецедента | ID: 6 |
| Краткое описание | Изменить данные существующего в базе данных договора энергоснабжения |
| Актеры, вовлеченные в прецедент | Главные актеры: работник договорного отдела  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Получены новые данные по существующему в базе данных договору |
| Фактические этапы прецедента | 1. Ввести измененные данные по договору |
| Состояние системы после окончания прецедента | Информация о договоре энергоснабжения обновлена |
| Альтернативные потоки | Нет |

**Таб. 7** Описание прецедента EnterTariff

|  |  |
| --- | --- |
| Имя прецедента | Прецедент EnterTariff |
| Идентификатор прецедента | ID: 7 |
| Краткое описание | Создать справочник, содержащий информацию о тарифах |
| Актеры, вовлеченные в прецедент | Главное актеры: экономист  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Получена информация о новых тарифах |
| Фактические этапы прецедента | 1. Проверить наличие тарифа в базе данных  2. Завести в справочнике информацию о новом тарифе |
| Состояние системы после окончания прецедента | Обновленная информация о существующих тарифах в базе данных |
| Альтернативные потоки | Изменить значения этого тарифа, если он найден в базе данных |

**Таб. 8** Описание прецедента EnterDataOfMeter

|  |  |
| --- | --- |
| Имя прецедента | Прецедент EnterDataOfMeter |
| Идентификатор прецедента | ID: 8 |
| Краткое описание | Ввести новые показания приборов учета в базу данных |
| Актеры, вовлеченные в прецедент | Главные актеры: расчетчик  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Получены новые показания приборов учета потребления электроэнергии |
| Фактические этапы прецедента | 1. Ввести новые показания прибора учета в соответствующую таблицу базы данных |
| Состояние системы после окончания прецедента | Обновленная информация о показаниях приборов учета потребления электроэнергии |
| Альтернативные потоки | Нет |

**Таб. 9** Описание прецедента ChargePrepayment

|  |  |
| --- | --- |
| Имя прецедента | Прецедент ChargePrepayment |
| Идентификатор прецедента | ID: 9 |
| Краткое описание | Выполнить авансовые начисления потребителям электроэнергии (контрагентам) |
| Актеры, вовлеченные в прецедент | Главные актеры: расчетчик  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Наступила определенная дата (скажем, 1-ое число каждого месяца) |
| Фактические этапы прецедента | 1. Проверить, было ли выполнено авансовое начисление для текущего контрагента  2. Выполнить авансового начисления для контрагента |
| Состояние системы после окончания прецедента | Информация о новом авансовом начислении для каждого контрагента введена в базу данных |
| Альтернативные потоки | Выдать соответствующее сообщение о том, что для текущего контрагента авансовое начисление уже было выполнено |

**Таб. 10** Описание прецедента ChargeBalance

|  |  |
| --- | --- |
| Имя прецедента | Прецедент ChargeBalance |
| Идентификатор прецедента | ID: 10 |
| Краткое описание | Выполнить итоговые начисления потребителям электроэнергии (контрагентам) |
| Актеры, вовлеченные в прецедент | Главные актеры: расчетчик  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Наступила определенная дата (скажем, 15-ое число каждого месяца) |
| Фактические этапы прецедента | 1. Проверить, было ли выполнено итоговое начисление для текущего контрагента  2. Выполнить итогового начисления для контрагента |
| Состояние системы после окончания прецедента | Информация о новом итоговом начислении для каждого контрагента введена в базу данных |
| Альтернативные потоки | Выдать соответствующее сообщение о том, что для текущего контрагента итоговое начисление уже было выполнено |

**Таб. 11** Описание прецедента FormPaymentDocuments

|  |  |
| --- | --- |
| Имя прецедента | Прецедент FormPaymentDocuments |
| Идентификатор прецедента | ID: 11 |
| Краткое описание | Сформировать платежные документы для всех контрагентов |
| Актеры, вовлеченные в прецедент | Главные актеры: расчетчик  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | В базе данных появилась информация о новых авансовых или итоговых начислениях |
| Фактические этапы прецедента | 1. Сформировать платежную ведомость для каждого контрагента  2. Отправить платежную ведомость контрагенту |
| Состояние системы после окончания прецедента | Каждый контрагент получил платежные документы за текущий месяц (год) |
| Альтернативные потоки | Нет |

**Таб. 12** Описание прецедента AccountPayment

|  |  |
| --- | --- |
| Имя прецедента | Прецедент AccountPayment |
| Идентификатор прецедента | ID: 12 |
| Краткое описание | Учет оплат, поступивших на счет компании |
| Актеры, вовлеченные в прецедент | Главные актеры: сотрудник по обработке оплат  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Информация о поступление оплат |
| Фактические этапы прецедента | 1. Определить, в счет какого договора она поступила, и привязать оплату к договору |
| Состояние системы после окончания прецедента | Обновленная информация в базе данных о связи оплат и договоров |
| Альтернативные потоки | Нет |

**Таб. 13** Описание прецедента ConductPayment

|  |  |
| --- | --- |
| Имя прецедента | Прецедент ConductPayment |
| Идентификатор прецедента | ID: 13 |
| Краткое описание | Произвести операцию разноски оплаты , т.е. связать поступившую оплату и счета потребителя |
| Актеры, вовлеченные в прецедент | Главные актеры: сотрудник по обработке оплат  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Информация о поступлении оплаты от какого-либо контрагента |
| Фактические этапы прецедента | 1. Определить, в счет какого договора поступила оплата  2. Определить обязательства внутри договора, на который поступила оплата |
| Состояние системы после окончания прецедента | Обновленная информация в базе данных о связи оплат и обязательств |
| Альтернативные потоки | Нет |

**Таб. 14** Описание прецедента CalculateDebt

|  |  |
| --- | --- |
| Имя прецедента | Прецедент CalculateDebt |
| Идентификатор прецедента | ID: 14 |
| Краткое описание | Рассчитать задолженности по каждому договору для каждого контрагента |
| Актеры, вовлеченные в прецедент | Главные актеры: сама система  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | По запросу пользователя (специалист по работе с задолженностью, сотрудник по обработке оплат), работающего с этой информацией |
| Фактические этапы прецедента | 1. Проверить, была ли рассчитана задолженность для текущего контрагента  2. Рассчитать задолженности для текущего контрагента по всем неоплаченным счетам |
| Состояние системы после окончания прецедента | Информация о существующих на данный момент задолженностях обновлена в базе данных |
| Альтернативные потоки | Обновить информацию (перерасчет задолженностей) для текущего контрагента |

**Таб. 15** Описание прецедента FormStructureOfDebt

|  |  |
| --- | --- |
| Имя прецедента | Прецедент FormStructureOfDebt |
| Идентификатор прецедента | ID: 15 |
| Краткое описание | Сформировать отчет, который выводит диаграмму о существующих задолженностях (текущая, просроченная, мораторная, реструктурированная, исковая) и их процентном соотношении |
| Актеры, вовлеченные в прецедент | Главные актеры: специалист по работе с задолженностью  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | По запросу отчет пользователя , работающего с этой информацией, сформировать отчет о ней |
| Фактические этапы прецедента | 1. Разбить задолженности на группы (текущая, просроченная, мораторная, реструктурированная, исковая)  2. Отобразить структуру задолженностей на диаграмме |
| Состояние системы после окончания прецедента | Сформирован отчет о структуре задолженностей на данное время |
| Альтернативные потоки | Нет |

**Таб. 16** Описание прецедента FormDynamicOfDebt

|  |  |
| --- | --- |
| Имя прецедента | Прецедент FormDynamicOfDebt |
| Идентификатор прецедента | ID: 16 |
| Краткое описание | Сформировать отчет, который выводит график или таблицу о динамике изменения задолженностей за прошедший месяц (как текущая задолженность изменилась за месяц) |
| Актеры, вовлеченные в прецедент | Главные актеры: специалист по работе с задолженностью  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | По запросу отчет пользователя, работающего с этой информацией, сформировать отчет о ней |
| Фактические этапы прецедента | 1. Разбить задолженности на группы (текущая, просроченная, мораторная, реструктурированная, исковая)  2. Отобразить динамику задолженностей на диаграмме |
| Состояние системы после окончания прецедента | Сформирован отчет о динамике задолжностей на данное время |
| Альтернативные потоки | Нет |

**Таб. 17** Описание прецедента PerformSalesAnalysis

|  |  |
| --- | --- |
| Имя прецедента | Прецедент PerformSalesAnalysis |
| Идентификатор прецедента | ID: 17 |
| Краткое описание | Выполнить интеллектуальная обработка существующих данных - анализ продаж поквартально, за год и тд |
| Актеры, вовлеченные в прецедент | Главные актеры: бизнес-аналитик  Второстепенные актеры: нет |
| Состояние системы до начала прецедента | Запрос информации об анализе продаж от руководства компании |
| Фактические этапы прецедента | 1. |
| Состояние системы после окончания прецедента | Пользователем получена информация об интеллектуальной обработке данных |
| Альтернативные потоки | Нет |