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**

КУРСОВАЯ РАБОТА

бакалавра направления 010400 «Информационные технологии»

**«Разработка веб-приложения для работы с биллинговой системой энергосбытовой компании»**

|  |  |
| --- | --- |
|  | Автор работы:  студент группы ВМИ-356  Барабанщикова Наталья Игоревна  Научный руководитель:  кандидат физ.-мат. наук,  доцент кафедры системного программирования  Радченко Глеб Игоревич  Оценка: \_\_\_\_\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_  Подпись: \_\_\_\_\_\_\_\_\_\_\_ |

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

10.02.2013

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы ВМИ-356 Н.И. Барабанщиковой,

обучающемуся по направлению 010400.62 «Информационные технологии»

1. **Тема работы**Разработка веб-приложения для работы с биллинговой системой энергосбытовой компании.
2. **Срок сдачи студентом законченной работы:** 31.05.2013 г.
3. **Исходные данные к работе (три основных источника)**
4. 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.;
5. Арлоу Д., Нештадт А. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование. – СПб.: Символ-плюс, 2007. – 624 с.;
6. Сьерра К., Бейтс Б. Изучаем Java. 2-е издание. – М.: Эксмо, 2012. -708 с.
7. **Перечень подлежащих разработке вопросов**
8. изучить возможности работы с базой данных Oracle Database 11g;
9. изучить существующие подходы разработки веб-приложений для работы с базой данных;
10. изучить технологии программирования Java SE, Java EE, ADF;
11. изучить среду разработки приложений JDeveloper 11g;
12. реализовать веб-приложение для работы с биллинговой системой энергосбытовой компании;
13. произвести тестирование веб-приложения.

**Дата выдачи задания:** «9» февраля 2013 г.

Научный руководитель Г.И. Радченко

Задание принял к исполнению Н.И. Барабанщикова

# Оглавление

[Оглавление 4](#_Toc351515285)

[Введение 5](#_Toc351515286)

[Актуальность работы 5](#_Toc351515287)

[Цели и задачи работы 5](#_Toc351515288)

[Структура и объем работы 6](#_Toc351515289)

[1. Обзор технологий построения информационных систем 7](#_Toc351515290)

[1.1 Технологии проектирования информационных систем 7](#_Toc351515291)

[1.1.1 Методы структурного проектирования ИС. 7](#_Toc351515292)

[1.1.2 Методы объектно-ориентированного проектирования ИС. 7](#_Toc351515293)

[1.2 Архитектура информационных систем 8](#_Toc351515294)

[1.3 Выбор СУБД для информационной системы 9](#_Toc351515295)

[1.4 Сервер приложений 10](#_Toc351515296)

[1.4.1 Oracle WebLogic 11g 10](#_Toc351515297)

[1.4.2 Sun GlassFish 10](#_Toc351515298)

[1.4.3 IBM Webspher 10](#_Toc351515299)

[1.5 Варианты построения клиентского приложения информационных систем 10](#_Toc351515300)

[1.5.1 Толстый клиент 10](#_Toc351515301)

[1.5.2 Тонкий клиент 11](#_Toc351515302)

[1.6 Варианты построения информационных систем с веб-интерфейсом 11](#_Toc351515303)

[2. Требования к веб-приложению биллинговой системой 14](#_Toc351515304)

[2.1 Функциональные требования 14](#_Toc351515305)

[2.2 Нефункциональные требования 14](#_Toc351515306)

[2.3 Диаграмма прецедентов 14](#_Toc351515307)

[3. Архитектура веб-приложения биллинговой системы 16](#_Toc351515308)

[3.1 Общий интерфейс системы 16](#_Toc351515309)

[3.2 Работник договорного отдела 16](#_Toc351515310)

[3.3 Экономист 18](#_Toc351515311)

[3.4 Расчетчик 19](#_Toc351515312)

[3.5 Сотрудник по обработке оплат 20](#_Toc351515313)

[3.6 Специалист по работе с задолженностью 21](#_Toc351515314)

[3.7 Бизнес-аналитик 21](#_Toc351515315)

[4. Реализация веб-приложения биллинговой системы 22](#_Toc351515316)

[5. Тестирование веб-приложения биллинговой системы 23](#_Toc351515317)

[Заключение 24](#_Toc351515318)

[Литература 25](#_Toc351515319)

[Приложение 1. Описание прецедентов 26](#_Toc351515320)

# 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 Oracle WebLogic 11g;
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, четырех разделов, заключения, библиографии и приложения. Объем работы составляет 34 страницы, объем библиографии – 10 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.

Глава архитектуры биллинговой системы содержит описание архитектуры системы. В этом разделе подробно рассмотрена общая архитектура системы, интерфейсы компонентов и детали их реализации.

Четвертая глава посвящена реализации биллиногово й системы при помощи технологий J2EE и Oracle ADF.

Глава тестирования …

В заключении сделаны выводы о проделанной работе.

Приложение содержит описание прецедентов для каждого пользователя системы …

# Обзор технологий построения информационных систем

## Технологии проектирования информационных систем

UML (Unified Modeling Language) [] – 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.

## СУБД для информационной системы

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 [] (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.

## Архитектура клиент-сервер

Клиент-сервер — архитектура вычислительной системы, в которой задачи или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами [ссылка] . 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 [ссылка].

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.

## 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 []. 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 Web Logic, 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

ЧТО ЗДЕСЬ ДОЛЖНО БЫТЬ?!

# Требования к веб-приложению биллинговой системы

## Функциональные требования

Энергосбытовая компания занимается продажей услуг физическим и юридическим лицам города Челябинска. С каждым контрагентом компания заключает договор купли-продажи, на основании которого осуществляется сделка между энергосбытовой компанией и потребителем. Данные о контрагентах и договорах хранятся в базе данных и могут быть отредактированы сотрудниками компании посредством использования биллинговой системы.

В соответствии с этим договором, компания каждый месяц выставляет контрагенту счет. Начисления формируются биллинговой системой автоматически в соответствии с показаниями счетчиков и тарифами, установленными компанией.

После выполнения начислений биллинговая система обрабатывает поступившие оплаты (соотносит оплаты с договорами) и вычисляет задолженности по каждому договору для контрагентов.

Анализируя данные о продажах услуг компании и задолженностях, биллинговая система формирует соответствующие отчеты, по результатам которых определяется политика компании в ее дальнейшей работе.

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, работающего с биллинговой системой энергосбытовой компании:

1. Система должна обеспечивать ввод, хранение и редактирование данных по контрагентам (физическим и юридическим) лицам.
2. Система должна обеспечивать ввод, хранение и редактирование данных по договорам энергоснабжения.
3. Система должна обеспечивать ввод данных по тарифам.
4. Система должна обеспечивать ввод показаний приборов учета.
5. Система должна осуществлять выполнение авансовых и итоговых начислений.
6. Система должна формировать платежные документы (счета).
7. Система должна осуществлять учет поступивших оплат.
8. Система должна осуществлять операцию разноски оплаты (формирования связей между начислением и оплатой)
9. Система должна подсчитывать задолженности по договорам.
10. Система должна осуществлять анализ структуры и динамики дебиторской задолженности.
11. Система должна осуществлять интеллектуальную обработку данных - анализ и прогнозирование продаж (поквартально, за год)

## 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 для работы с биллинговой системой энергосбытовой компании были зафиксированы следующие нефункциональные требования:

1. Система должна быть разработана на программном обеспечении Oracle.
2. Система должна быть реализована в соответствии с трехзвенной клиент-серверной архитектурой: клиент (стандартный веб - браузер) - сервер приложений - сервер БД.

## Прецеденты использования биллиноговой системы

Можно выделить шесть основных актеров, взаимодействующих с разрабатываемой биллинговой системой ():

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

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

*Анализ продаж* – 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.

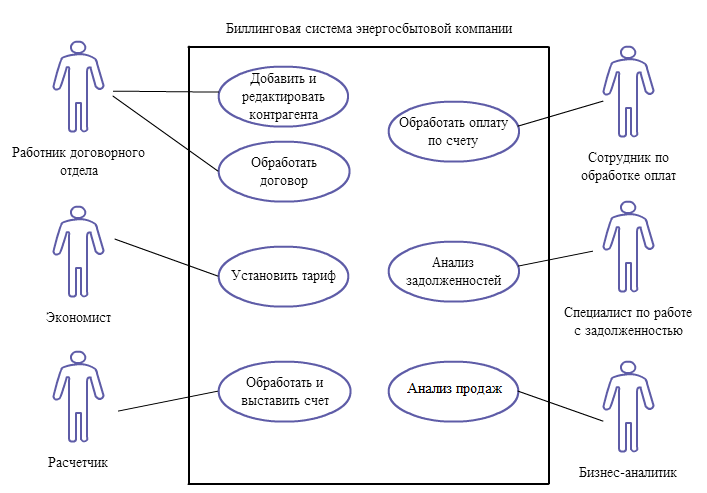
****

Рис. 1. Диаграмма прецедентов биллинговой системы   
энергосбытовой компании

# 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.

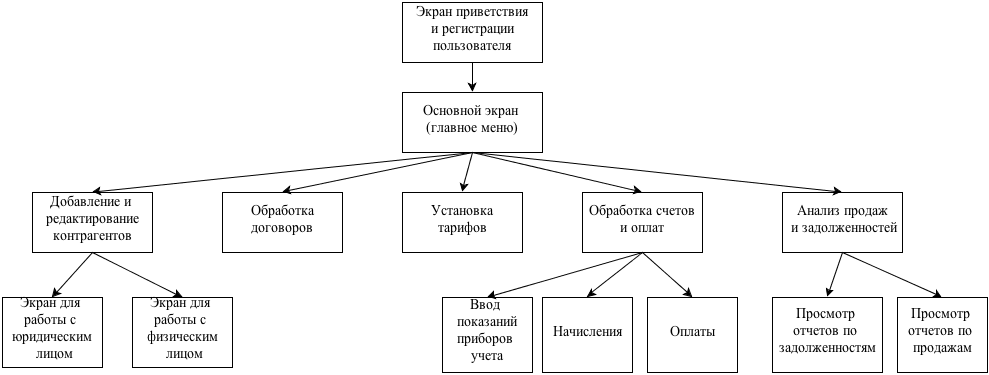


Рис. 2. Диаграмма доступных пользовательских интерфейсов

## Работник 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).

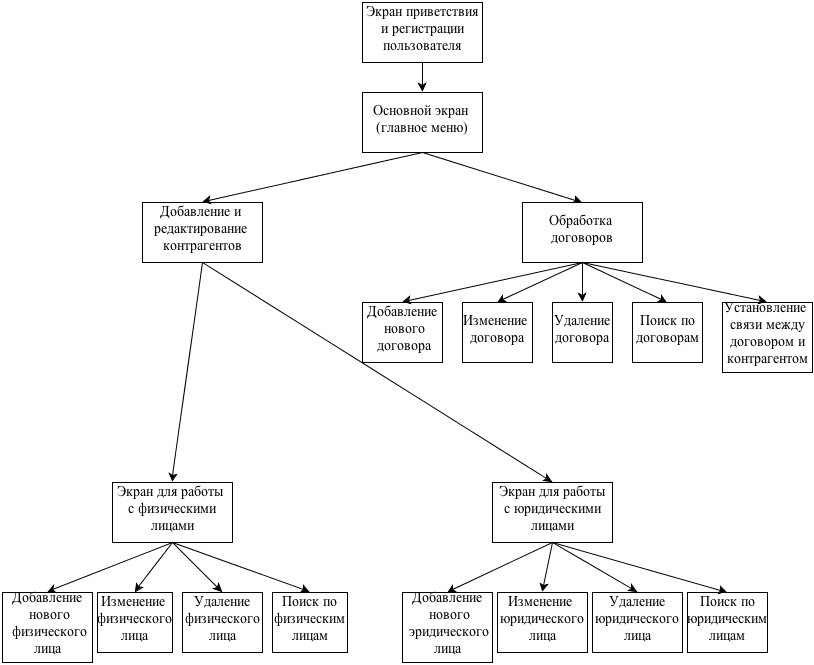


Рис. 3. Возможности работы с системой для работника договорного отдела

## Экономист

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).

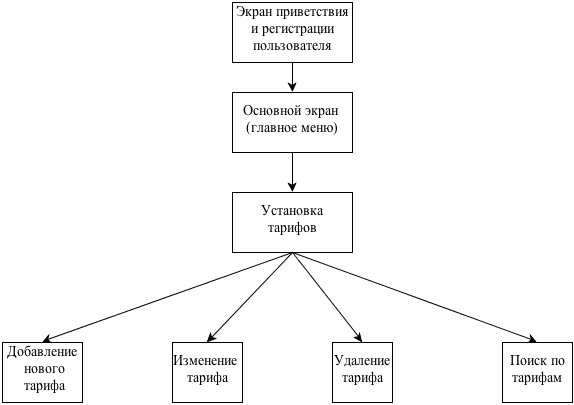


Рис. 4. Возможности работы с системой для экономиста

## Расчетчик

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).

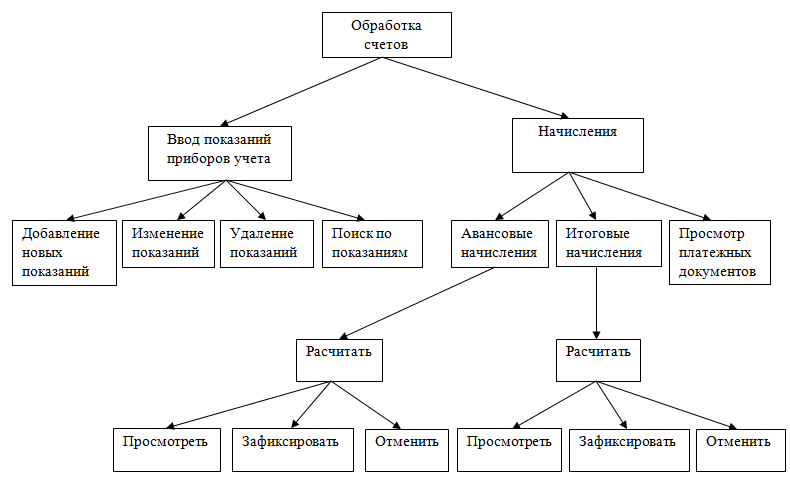


Рис. 5. Возможности работы с системой для расчетчика

## Сотрудник 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).

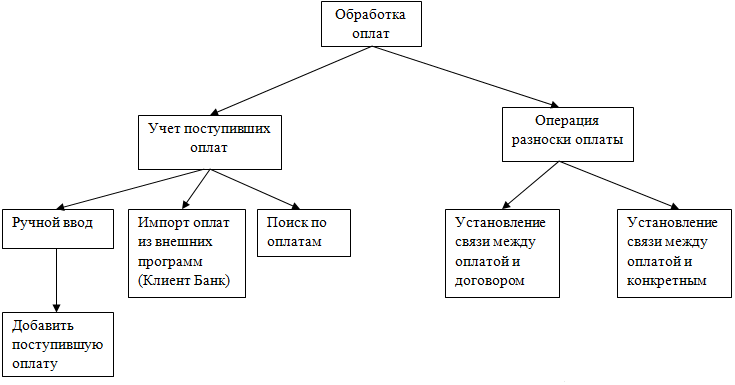


Рис. 6. Возможности работы с системой для сотрудника по обработке оплат

## Специалист 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).

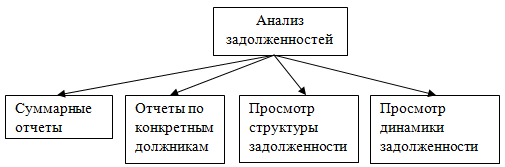


Рис. 7. Возможности работы с системой для специалиста по работе с задолженностью

## 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).

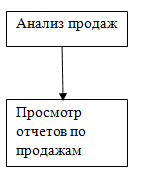


Рис. 8. Возможности работы с системой для бизнес-аналитика

# Реализация веб-приложения биллинговой системы

КАКИЕ ПУНКТЫ?!

База данных, роли пользователей и ограничения на доступ к функциям по ролям, хранение паролей…

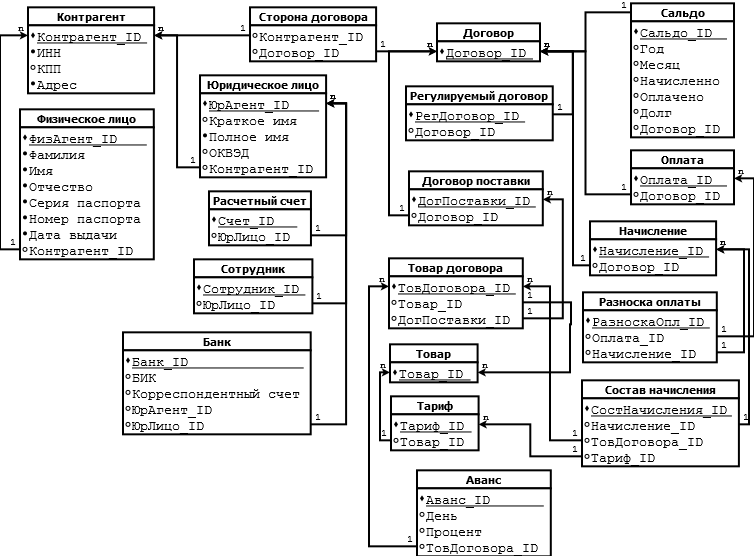


Рис. 9. Схема базы данных

# Тестирование веб-приложения биллинговой системы

Скрин-шоты форм

# Заключение

# 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 |