ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЗИРОВАННАЯ СИСТЕМА РАСЧЕТА РАСХОДОВ МАТЕРИАЛОВ И ЗАТРАТ ДЛЯ ФИЛИАЛА «КУРСК ЭНЕРГОЗАЩИТА»

Выпускная квалификационная работа
обучающегося по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения,
группы 07001402
Игнатова Дмитрия Александровича

Научный руководитель

к.т.н., доцент Чашин Ю.Г

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Развитие информационных технологий в 21 веке подразумевает в себе полную компьютерную автоматизацию во всех сферах деятельности человека, в это прогрессивное время – добивается успеха тот, кто идет нога в ногу с технологиями, внедряет их в свое производство, обучает взаимодействию с компьютерными информационными системами свой персонал, переводит документацию и бухгалтерию в электронный вид.

Актуальность работы заключается в внедрении современных информационных технологий и исключении человеческого фактора при расчетах.

Целью данной дипломной работы является реализация автоматизированной системы расчета расхода материалов и затрат в ОАО «Фирма Энергозащита», филиал «Курскэнергозащита» и дальнейшее ее внедрение.

Основными задачами являются: реализовать возможность осуществлять контроль за работниками, их обязанностями, составление списка оказываемых услуг и составлению смет согласно этим услугам, определению расходуемых материалов, необходимых для выполнения работ, определить эффективность организации, согласно количеству заказов, процентной прибыли по сделке, а также расходов на осуществления этих работ: материалов и заработной платы сотрудникам. Расчет заработной платы будет производиться автоматически, исходя от того, какую должность имеет рабочий и какой фронт обязанностей он может совмещать. Интерфейс реализованной программы обязан быть максимально прост в эксплуатации, доступ к должно иметь только начальство, для этого должна быть предусмотрена система авторизации.

Для того, чтобы достигнуть вышеперечисленных задач – следует выделить и указать ряд ключевых компонентов, который включает в себя:

1) Произвести краткий анализ деятельности организации, для которой будет реализована информационная систем

2) Согласно анализу деятельности организации, прийти к выводу, какой функционал должна иметь информационная система;

3) Произвести обзор существующей информационной системы и выявить преимущества и недостатки;

4) Определить инструментарий для реализации системы;

5) Проектировка и практическая реализация базы данных;

6) Проектировка модулей авторизации и администрирования, а также дальнейшая их реализация;

7) Апробация и дальнейшее внедрение информационной системы

Данная выпускная квалификационная работа содержит в себе введение, 3 главы, заключение, список литературы и приложение.

Введение содержит в себе краткую информацию о работе, определяет основные цели и задачи, которые необходимо реализовать и средства достижения данных целей и задач.

Первая глава информирует о особенностях и производит анализ организации, в которую необходимо внедрить информационную систему, рассмотрение существующего аналога и его краткий анализ, а также устанавливаются задачи, которые необходимо решить.

Вторая глава посвящена проектированию автоматизированной системы, в частности: обоснование выбора СУБД, программной оболочки. Отталкиваясь от этого выбора, происходит разработка приложения и его графического интерфейса.

Третья глава описывает основной функционал реализованной автоматизированной системы, ее пользовательский интерфейс, а также рассматриваются основные модули.

В заключении подводится итог выполненной работы.

Дипломная работа состоит из 56 страниц, 35 рисунков, 1 таблицу, 7 листингов и приложения включающего 10 страниц.

Глава 1. ПОСТАНОВКА ЗАДАЧИ

1.1 Особенности организации

14 апреля 1966 года – начало истории открытого акционерного общества «Фирма Энергозащита», которое обладает уникальным опытом и мощной ресурсной базой в производстве теплоизоляционных материалов, а также разветвленной филиальной сетью по выполнению услуг в сфере проектирования, выполнения теплоизоляционных и антикоррозионных работ. Сегодня это компания - 137 производственных площадок, 1500 - реализованных проектов, как в России, так и за рубежом, состоящая из 2800 высококвалифицированных специалистов. Большая часть объектов, на которых производятся работы – атомные электростанции, заказчиком в данном случае выступает государственная корпорация по атомной энергетике «Росатом».

В связи с тенденцией работ на атомных станциях и их расположении в городах, которые находятся, в большинстве своем, далеко от офисов организации – сотрудники работают в командировочном режиме. С этой целью выделяются специальный транспорт для доставки рабочих в город, где происходит строительство на бесплатной основе, по приезду – мастера обязаны обеспечить жильем свои бригады, для этого арендуются квартиры в относительной близости к объекту работы. Каждому командированному сотруднику предусмотрена сумма в размере 300 рублей в день, так называемые «командировочные». Помимо этого, два раза в день отправляются частные маршрутные автобусы, целью которых является доставить непосредственно от места расселения рабочих до контрольно-пропускного пункта.

В связи с квалификацией рабочих и их специализацией – ставиться задача на выполнение: теплоизоляция, монтажные работы, работа по металлу, существуют отдельные таксы для работы на высоте или вблизи мест, где

повышен радиоактивный фон. В организации предусмотрен шестидневный рабочих день, однако суббота оплачивается по двойному тарифу.

За выполнением рабочих стандартов наблюдает мастера, они обязаны произвести анализ качества выполненной работы и записать объемы выработки, плюсы и минусы деятельности рабочих. Все записанные данные мастер обязан передать начальнику участка, который на свое усмотрение может произвести вторичную проверку качества выполнения работы, после чего подсчитать сумму, на которую выполнена работа и определить сумму сдельной зарплаты сотрудника, в зависимости от выполненного объема работы, условий работы и процента брака. Составленные документы отправляются в центральный офис, который согласует полученные цифры заработной платы и анализирует коэффициент прибыли за определенный период работы и согласно этому определяется необходимость премировать или штрафовать мастера или начальника участка.

Согласно полученный выше сведениям, организация имеет широкий спектр задач, которые необходимо автоматизировать. В частности, необходимо:

1) Создание списка сотрудников, в котором можно получить информацию о должности и разряде рабочего, квартиры на которой он проживает, сумме аренды за данную жилищную площадь. Список должен быть реализован в виде таблицы.

2) Возможность ввести электронную документацию, в которой будет описан список представляемых услуг и цена на произведение этих работ. Аналогично каждой услуге будет составлен перечень расходуемых материалов, конкретно необходимых под определенную работу и подсчитана цена за них.

3) Предоставление возможности начальнику участка составлять сметы, согласно услугам, которые оказывает организация. Смета, помимо выполненной услуги будет содержать в себе данные о том, какие рабочие участвовали в этих работах, даты начала и выполнения работы, объект на

котором она будет осуществлена, и кто является заказчиком, определен коэффициент прибыли по смете, чтобы грамотно построить процесс получения положительного притока финансов.

4) Для того, чтобы исключить человеческую ошибку в подсчетах заработной платы – составить алгоритм, который будет анализировать работу сотрудников, согласно сметы и распределять между ними сумму, которая заложена в смете для рабочих, отталкиваясь от объема выработки каждого сотрудника и его квалификационный уровень.

5) Каждый из реализованных списков должен иметь возможность редактирования, добавления и удаления записей. Для удобства поиска данных – необходимо создать поиск по каждому из полей таблицы и возможность сортировать данные по алфавиту.

1.2 Обзор и анализ существующих аналогов

Для того, чтобы спроектировать удачную автоматизированную систему, необходимо проанализировать существующие аналоги и перенять опыт или выявить недостатки, которые необходимо исправить в собственном приложении. Реализуемый нами ресурс имеет структуру административного приложения, поэтому увидеть подобные аналоги очень сложно, так как большинство из них скрыты от внешних пользователей аутентификацией. В организации, для которой будет создана автоматизированная система, уже есть свой онлайн – ресурс, который имеет частично выполнять схожий объем услуг, однако, в связи с развитием информационных технологий – данная информационная система устарела и было принято решение о создании новой. Ниже будет представлен анализ этого ресурса.

Система 1. Сайт организации «Энергозащита»

Так как реализованное приложение в процессе дипломной работы будет настроено под определенный филиал, для сравнения будет продемонстрировано существующее раннее приложение, находящееся по

адресу: http://www.egz.ru. На рисунке 1.1 можно увидеть главную страницу сайта и проанализировав ее внешний вид, можно сделать вывод, что дизайн сайта устарел, его графическое оформление непригодно для долгого пользования, в связи с неграмотно подобранной цветовой гаммой.

Рис. 1.1. Главная страница

В качестве страницы, которая требует автоматизации, рассматривается тот раздел информационного портала, на котором находится список услуг, которые производит организация.

Рис. 1.2. Главная страница

Как мы можем видеть, в существующем списке оказываемых услуг очень кратко изложена информация о наименовании работ и материалы, необходимые для этого. Так как эта страница представлена статически, то никакого редактирование и хранение в базе данные здесь не предусмотрено.

Для того, чтобы определить цену на услуги, была реализована специальная страница, пример которой можно увидеть на рисунке 1.3.

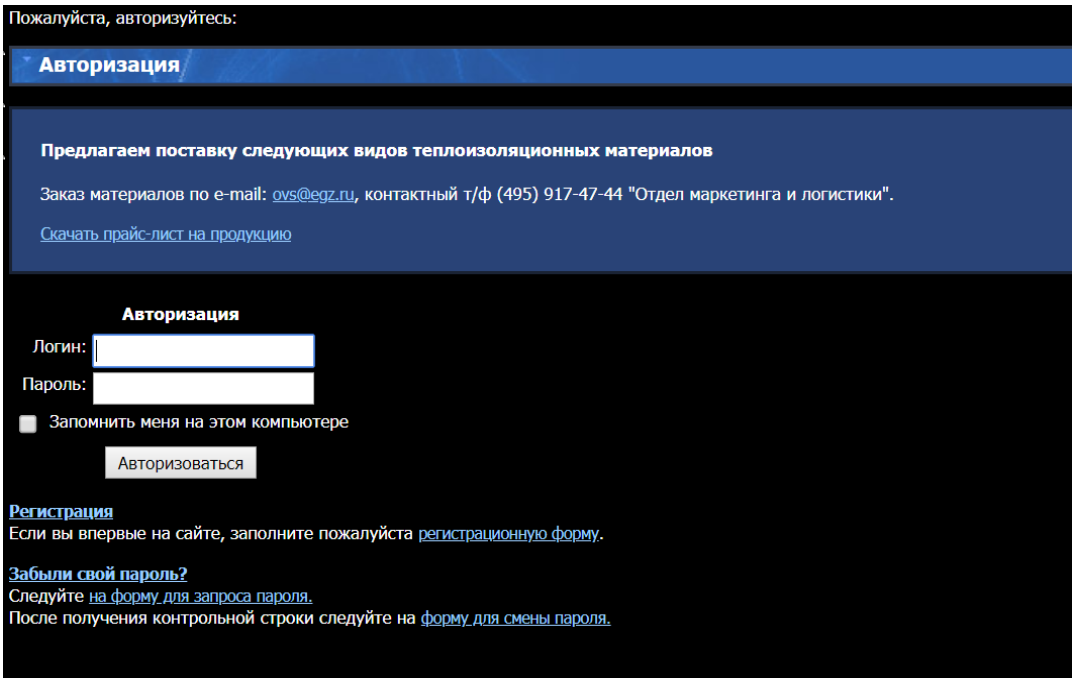

Рис. 1.3. Прайс - лист

На сайте существует авторизация, для того, чтобы можно было оставлять сообщения администратору сайта. Однако, в связи с плохим обслуживанием сайта, большинство модулей не работает, в том числе регистрация. В связи с этим, обязательным фактором является автоматизация отображения цен на услуги и расчет итоговой цены. На странице предлагается скачать excel файл, который содержит данные за 2005 год. Сама концепция хранения данных в формате excel таблицы и скачивание ее с целью ознакомления с ценами – крайне устаревшая, это неудобно ни для работников, которые должны будут заново составлять таблицу и загружать ее на сервер,

так и для клиентов, которым необходимо что – то дополнительно устанавливать на свой компьютер.

Проанализировав все страницы, мы можем прийти к выводу по поводу того, что необходимо исправить при реализации новой автоматизированной системы:

1) Создание рабочего модуля авторизации, регистрация должна быть упразднена и аккаунты выдаваться сотрудникам от администратора.

2) Автоматизировать список оказываемых услуг и прайс – листа для них, переведя в список в виде таблицы, что позволит избавится от лишних excel файлов и установки дополнительного софта.

Сегодня одной из самых распространенных программ по автоматизации бухгалтерского учёта является «1С: Бухгалтерия». Приобретая программы семейства 1С, важно чтобы была проведена правильная установка и настройка 1С, а также в некоторых случаях может понадобиться вводный консультационный курс. Графический интерфейс приложения продемонстрирован на рисунке 1.4.

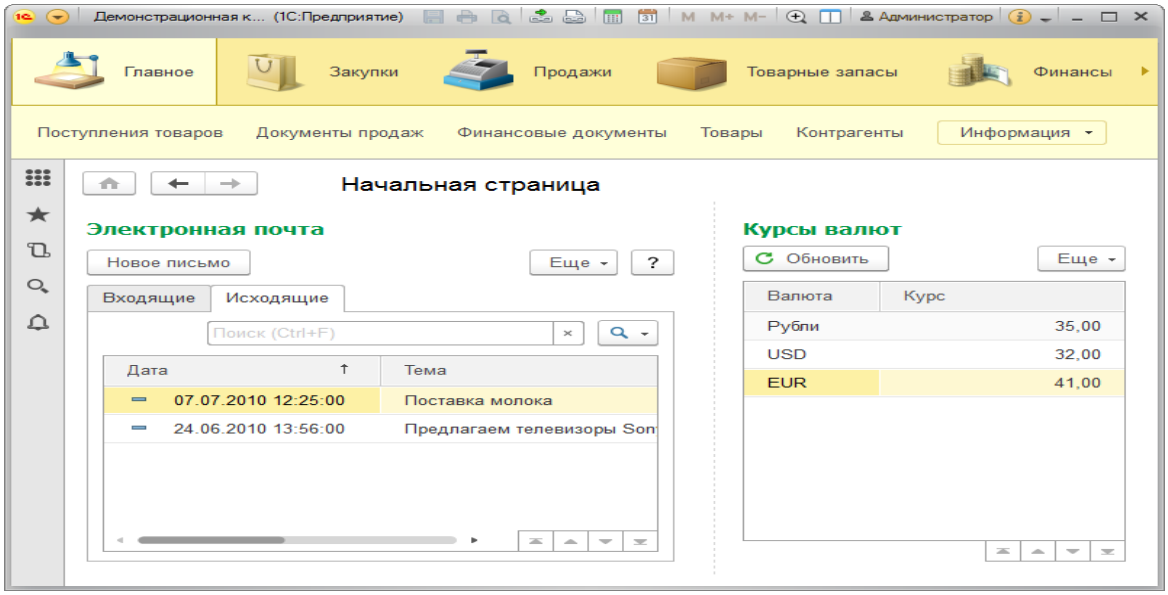

Рис. 1.4. «1С: Предприятие»

Однако не смотря на то, что программа получила широкое распространение, отношение к ней у специалистов неоднозначное. Одни

считают её самой лучшей среди всех существующих средств автоматизации бухгалтерского и налогового учёта, другие же находят в ней множество недостатков и недоработок [1].

На самом деле все зависит от предприятия, на котором используется данная программа. Для одних это оптимально подходящий вариант, но для других — 1С не в состоянии решить все поставленные задачи. Поэтому прежде чем начать работать с программой, важно оценить все её достоинства и недостатки.

Вот некоторые положительные характеристики «1С: Бухгалтерии».

1.Данная программа позволяет вести все существующие виды бухгалтерского и налогового учета.

2.«1С: Бухгалтерия» универсальная программа, которая может использоваться на самых разных предприятиях. Она основана на платформе «1С: Предприятие» и её можно использовать под конкретные нужды бизнеса. Таким образом, с помощью данной программы возможно решение самых различных вопросов.

3.С помощью последней версии «1С: Бухгалтерия 8» возможно решение даже самых сложных задач, благодаря её высокой производительности.

4.В нашей стране регулярно происходят изменения в законодательстве связанном с ведением бухгалтерского и налогового учета. Разработчики «1С: Бухгалтерии» внимательно отслеживают малейшие изменения и своевременно обновляют формы отчётности в программе.

Однако в «1С: Бухгалтерии» существуют и некоторые недостатки.

1.Чаще всего для того чтобы программа решила все поставленные перед ней задачи, её необходимо дорабатывать в соответствии с конкретными требованиями предприятия.

2.В процессе перехода с другой бухгалтерской программы на «1С: Бухгалтерию» не редко возникают трудности с переносом всей информации. Случается, так, что большую часть информации приходится переносить вручную.

3.«1С: Бухгалтерия» — сложная программа, требующая специального обучения.

Очень важно учитывать все нюансы при выборе программы для ведения бухгалтерского и налогового учёта. Необходимо подобрать тот продукт, который будет максимально соответствовать требованиям конкретного бизнеса.

Галактика ERP — автоматизированная система управления, позволяющая в едином информационном пространстве оперативно решать главные управленческие задачи, а также обеспечивать персонал предприятия различного уровня управления необходимой и достоверной информацией для принятия управленческих решений [2]. Интерфейс приложения нельзя назвать очень удобным и внешне он похож на excel (см. рис. 1.5).

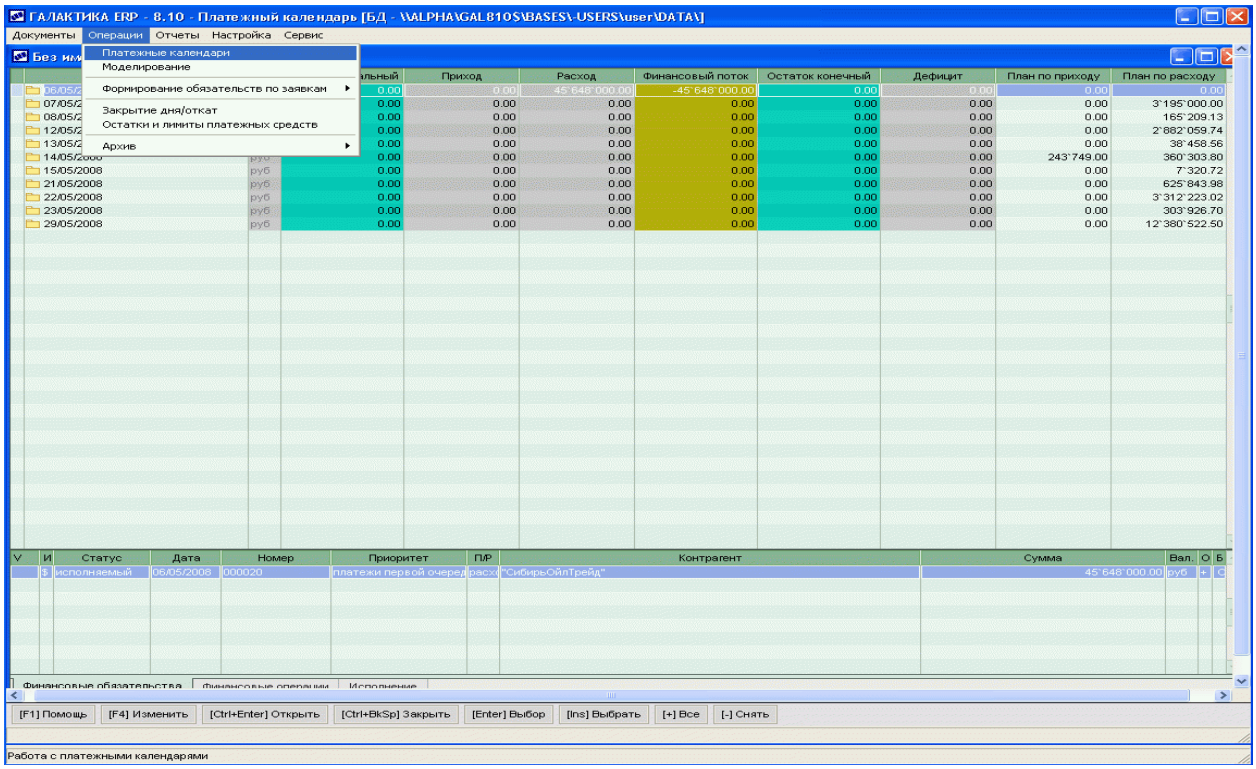

Рис. 1.5. Галактика ERP

Достоинства системы «Галактика».

Система имеет очень широкий набор функций, по-видимому, самый большой среди российских систем и покрывает широкий спектр запросов

Заказчиков. Это единственная система, которая имеет функции планирования и производства.

В бухгалтерской части система построена в полном соответствии с представлениями бухгалтеров об автоматизированной системе. Имеет полный набор стандартной и специальной бухгалтерской отчетности.

Во всех своих модулях система очень хорошо обеспечивает нужды печати оперативных документов (накладных, счетов-фактур, счетов, сопроводительных документов и т.д.). Это обеспечивает достаточную эффективность ее использования.

Система имеет достаточно много параметров настройки на особенности конкретного Заказчика.

Система имеет очень простые, эффективные и универсальные средства расширения форм ввода и определения новых справочников. Генераторы финансовой и табличной отчетности очень эффективны и просты.

Система уже отлажена (в своем ядре) и, по-видимому, достаточно устойчиво работает.

Имеет четкую стратегию и тактику продвижения системы на рынке, а также развития системы.

Недостатки системы «Галактика».

Несмотря на заявленную, на первой странице своего описания, правильную цель работы предприятия и задачу внедрения системы на нем, реально галактика не обеспечивает выполнение этой цели. система не является управляющей. она не реализует алгоритмов формирования оптимальных запросов на производство и/или снабжение в зависимости от состояния спроса, планов, прогнозов или их комбинации. внедрение ее не приносит конкретной прибыли.

Система не имеет механизма определения и контроля процедур выполнения конкретных операций или группы операций (например, определение процедуры снабжение: способ формирования заявки - заявка - выбор поставщика - формирование заказа - отслеживание его выполнения -

процедура получения на склад), что не позволяет руководителю быть уверенным, что его управляющие решения исполняются.

Система не имеет функций, необходимых для обеспечения деятельности крупных корпораций (централизованное снабжение, распределение функций между организациями, передача полномочий от одной организации к другой, взаиморасчеты внутри корпорации и т.д.)

Система, практически, не является интегрированной. Большинство модулей практически не связано между собой, а их связь с финансами очень условна, т.к. документы в финансовом модуле вводятся вручную на основании первичных документов, что приводит к расхождению в материальном и финансовом учете.

«Парус» - комплексная система, которая позволяет организовать бухгалтерский учет в полном объеме, учет основных фондов, материалов и МБП, рассчитывает заработную плату и имеет мощную аналитику.

В настоящее время выпущена версия Парус 8 на платформе Oracle, которая переводит данную систему в разряд корпоративных систем. Графический интерфейс выглядит более лаконично, чем «Галактика» (см. рис.1.6).

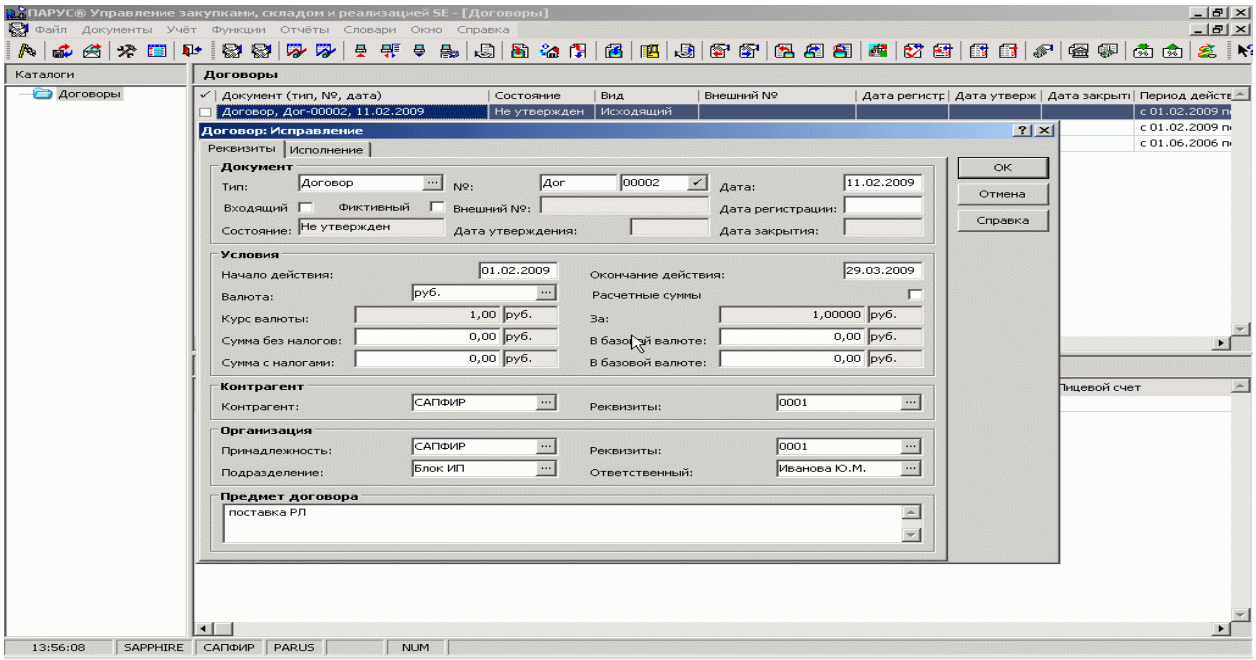

Рис.1.6. Комплексная система «Парус»

Система работает соответственно Международных стандартов бухгалтерского учета, предоставляя возможность проводить учет по МСБУ, и одновременно поддерживать любую трансформацию данных, которые накапливаются в учетных регистрах, к одному виду учета к другому. Можно выделить следующие задачи.

1. Автоматизирование бухгалтерских операций в тех организациях, где источниками финансов выступает собственный капитал, и капитал, который привлекается в предприятие.
2. Оптимизация сложных бизнес-процессов, и строение учета «от начального документа».
3. Полная поддержка многих пользовательских видов учета, и обеспечение любого преобразования данных, накопленных в документации по учету, в разных видах учёта.
4. Оперативное решение типичных и нетипичных заданий по бухгалтерскому учету, в режиме «on-line».
5. Контроль по наличию и движению имущества. Также рассматривается рациональное использование производственных ресурсов, своевременное предупреждение негативных факторов в финансовой деятельности.
6. Расчет данных в приложении «Расчет зарплаты», для создания учетного журнала, формирования «Кассовых документов».

Помимо явных преимуществ программы «Парус» существуют также некоторые недостатки и неудобства. Например, Парус 4х по своей сути закрытая система и может быть изменена пользователем. Только разработчики имеют право проводить модификацию базовых модулей и приспосабливать их к специфике конкретного предприятия. Такой процесс достаточно дорого стоит, и часто вызывает трудности обновления версии. Также стоимость этой программы определяется стоимостью одного рабочего места, и с увеличением количества пользователей соответственно растёт. Так, при количестве пользователей от 20 до 40 человек, стоимость может быть достаточно высокой.

Кроме цены, освоение программы также может вызывать некоторые трудности, потому как правильно работать там смогут только высококвалифицированные работники [3].

В итоге исследования основных аналогов приложений, выполняющих схожий персонал, были определены недостатки и преимущества, которые следует учесть при разработке автоматизированного приложения.

1.3 Требования к автоматизированной системе

Согласно современным тенденциям, мы обязаны представить определенный список функциональных характеристик.

1) Автоматизированная система обязана обеспечивать выполнения перечисленных функций:

- возможность осуществлять авторизацию;
- после авторизации, в приложении должны выводиться данные пользователя, его имя и фотография;
- все данные, хранимые в приложении, выводятся в виде таблицы;
- динамического хранения данных, каждая таблица должна иметь возможность дополняться, каждое должно иметь возможность редактирования или удаления, при необходимости.
- экспорт полученных таблиц в формат excel, pdf файла, для дальнейшего вывода на печать.

2) Представление требований, в каком виде необходимо хранить входные данные.

После того, как пользователь введет в окне браузера адрес ресурса, и ему будет предложена возможность авторизоваться, входными данные будут:

- логин;
- пароль.

Так как процесса регистрации не будет, в связи с тем, что приложение рассматривается, как административное, то пароль и логин будет отправлен по электронной почте от разработчика ресурса.

Авторизованный пользователь будет иметь возможность вносить изменения в каждую из таблиц автоматизированной системы, входные данные будут изменяться согласно тому, с какой из таблиц будет произведена работа.

Персонал, получивший право администрировать приложение, на странице – управление сотрудниками, имеет возможность вносить такие данные как:

- ФИО сотрудника;
- должность;
- адрес проживания;
- арендная плата за съемную квартиру.

Персонал, получивший право администрировать приложение, на странице – управление расходуемыми материалами, имеет возможность вносить такие данные как:

- название материала;
- цена материала.

Персонал, получивший право администрировать приложение, на странице – оказываемые услуги, имеет возможность вносить такие данные как:

- название услуги;
- цена за метр;
- площадь.

Персонал, получивший право администрировать приложение, на странице – материалы для услуг, имеет возможность вносить такие данные как:

- внешний ключ расходуемого материала;
- внешний ключ оказываемой услуги;
- количество материалов для услуги.

Персонал, получивший право администрировать приложение, на странице – услуги рабочих, имеет возможность вносить такие данные как:

- внешний ключ сотрудника;
- внешний ключ оказываемой услуги.

Персонал, получивший право администрировать приложение, на странице – список заказчиков имеет возможность вносить такие данные как:

- юридическое название заказчика;
- мобильный телефон заказчика.

Персонал, получивший право администрировать приложение, на странице – сметы имеет возможность вносить такие данные как:

- внешний ключ заказчика;
- объект, где производятся работы;
- город, где находится объект;
- процентная прибыль сметы;
- дата начала работ;
- дата конца работ.

Персонал, получивший право администрировать приложение, на странице – смета услуг имеет возможность вносить такие данные как:

- внешний ключ таблицы услуги рабочего;
- внешний ключ из таблицы смет.

Персонал, получивший право администрировать приложение, на странице – смета услуг имеет возможность вносить такие данные как:

- внешний ключ таблицы услуги рабочего;
- внешний ключ из таблицы смет.

Персонал, получивший право администрировать приложение, на странице – вычисление заработной платы имеет возможность вносить такие данные как:

- внешний ключ из таблицы управления сотрудниками;
- премия;

- штраф;
- дата выплаты заработной платы.

Поле – к зачислению имеет возможность автоматического расчета, поэтому никаких входных данных для этого не нужно.

Для каждой из таблиц необходимо реализовать ввод входных данных так, чтобы это было максимально удобно и понятно: данные должны вводиться в специальные поля формы, в случаях, когда происходит работа с добавлением данных в таблицу, в которой имеются внешние ключи, необходима возможность их выбора через выпадающий список. Редактирование должно быть предусмотрено при нажатии на определенную строку таблицы в следствии чего отображается всплывающее модальное окно, в котором выводится форма, для удаления в приложении необходимо реализовать специальную кнопку- ссылку.

3) Требования к серверу

Для того, чтобы ресурс осуществлял свою работу, его необходимо перенести на специальную площадку, называемую сервером, к которому должны иметься определенные требования:

- круглосуточная работа;
- широкий интернет - канал;
- удобный доступ для разработчика автоматизированной системы.

4) Требование к обслуживающему персоналу

Нет смысла создавать какую-либо автоматизированную систему, если она не будет обслуживаться специалистами, необходимо сообщать о каждом сбое системы, багах, для дальнейшего их устранения соответствующим персоналом. Необходимо учесть тот факт, что приложение имеет локальный масштаб, поэтому персонал можно ограничить одним – двумя программистами.

5) Требования к составу и параметру технических средств

Чтобы пользоваться приложением необходим персональный компьютер, или специальное компьютерного оборудование, которое имеет доступ к интернету.

6) Требование к программным средствам и среде разработки

Так как технологии двигаются семимильными шагами, важно определить, на какой платформе будет производится разработка приложения, поэтому стоит выбрать Yii2 Framework, который дает возможность использовать технологию MVC, основными языками программирования выбраны – PHP и Javasctipt, а также библиотека AJAX jQuery для осуществления асинхронных запросов, для разметки и оформления страниц используются HTML и CSS. В качестве среды программирования был выбран PhpStorm от компании JetBrains, которая позволяет удобный интерфейс для полной автоматизации работы с различными файлами.

7) Требование к графическому оформлению

Дизайн автоматизированной системы необходимо спроектировать таким образом, чтобы управление ним было максимально удобно на протяжении большого количества времени, цвета подобраны таким образом, чтобы человеческий глаз не уставал от яркости или тусклости гаммы. Для перехода между страницами, в которых будут находится соответствующие таблицы, требуется реализовать вертикальное меню.

8) Требования к программным средствам, используемым программой

У клиента, для работы с автоматизированной системой должна быть установлена любая операционная система, поддерживающая взаимодействие с интернетом и работу современного веб браузера. Так же должен быть установлен сам браузер с поддержкой Javascript и стандарта HTML5.

Глава 2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Выбор средств разработки

В процессе планирования разработки автоматизированного приложения встал вопрос о том, какие программные средства взять за основу, в нынешнее время выбор настолько обширен и у каждого языка программирования и его среды есть преимущества и недостатки, поэтому в качестве серверного языка был выбран PHP, и платформой для всего этого выступил YII2 framework, в качестве среды программирования в данной ситуации наиболее удачным выбором является PhpStorm от компании JetBrains. Такое решение было принято неспроста, оно обусловлено тем, что framework YII2, как и большинство самым передовых фреймворков разделяет данные приложения, пользовательского интерфейса и управляющей логики на три компонента, такая технология имеет название MVC.

Это высокоэффективный, основанный на компонентной структуре PHP-фреймворк для быстрой разработки крупных веб-приложений. Он позволяет максимально применить концепцию повторного использования кода и может существенно ускорить процесс веб-разработки. Название Yii (произносится как Yee или [ji:]) означает простой (easy), эффективный (efficient) и расширяемый (extensible). Превосходство Yii над другими фреймворками заключается в эффективности, широких возможностях и качественной документации. [4]

Отдельным немаловажным фактором является выбор СУБД, на основе которого будет построено приложение. Оценив преимущества и недостатки различных СУБД был выбран PostgreSQL. Это не просто реляционная, а объектно-реляционная СУБД, что даёт ему некоторые преимущества над

другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Фундаментальная характеристика объектно-реляционной базы данных - это поддержка пользовательских объектов и их поведения, включая типы данных, функции, триггеры, домены и индексы. Это делает данную СУБД невероятно гибкой и надежной.

Функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса. Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие [5].

Триггеры определяются как функции, инициируемые DML—операциями. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования.

В PostgreSQL имеется поддержка индексов следующих типов: B-дерево, хэш, R-дерево, GiST, GIN. При необходимости можно создавать новые типы индексов . Индексы в PostgreSQL обладают следующими свойствами:

- возможен просмотр индекса не только в прямом, но и в обратном порядке — создание отдельного индекса для работы конструкции ORDER BY ... DESC не нужно;
- возможно создание индекса над несколькими столбцами таблицы, в том числе над столбцами различных типов данных;

- индексы могут быть функциональными, то есть строиться не на базе набора значений некоего столбца/столбцов, а на базе набора значений функции от набора значений;
- индексы могут быть частичными, то есть строиться только по части таблицы (по некоторой её проекции); в некоторых случаях это помогает создавать намного более компактные индексы или достигать улучшения производительности за счёт использования разных типов индексов для разных (например, с точки зрения частоты обновления) частей таблицы;

PostgreSQL может обрабатывать много данных. Текущие опубликованные ограничения перечислены на таблице 1:

Таблица 2.1

Ограничения в размерах данных в PostgreSQL

Максимальный размер базы данных	Неограничен
Максимальный размер таблицы	32 TB
Максимальный размер строки	1.6 TB
Максимальный размер поля	1 GB
Максимальное количество строк в таблице	Неограниченно
Максимальное количество столбцов в таблице	250-1600 в зависимости от типа столбца

2.2 Технология построения веб сайта

На этапе выбора технологии построения веб – приложение, итоговое предпочтение было отдано MVC не с проста. Шаблон проектирования MVC предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

В приведённом определении под компонентом следует понимать некую отдельную часть кода, каждая из которых играет одну из ролей Контроллера, Модели или Представления, где Модель служит для извлечения и манипуляций данными приложения, Представление отвечает за видимое пользователю отображение этих данных (то есть, в применении к вебу, формирует отдаваемый сервером браузеру пользователя HTML/CSS), а Контроллер управляет всем этим оркестром [6].

Классическая схема веб – приложения рассматривается на рисунке 2.1.

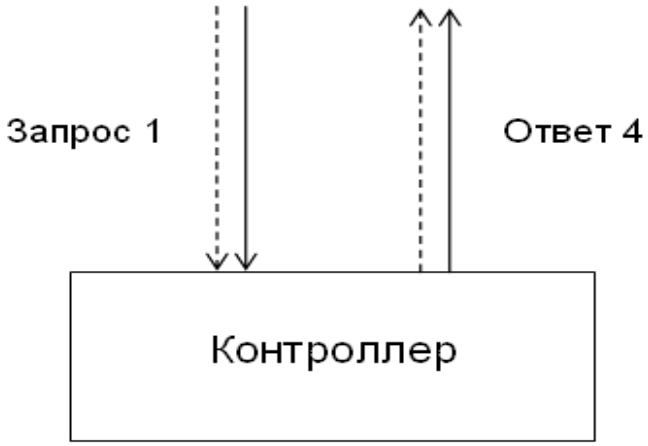

Рис. 2.1. Схема приложения с использованием MVC

На этом и последующем рисунках пунктирными линиями показана управляющая информация (такая, например, как ID запрашиваемой записи блога или товара в магазине), а сплошными – собственно данные приложения (которые могут храниться в БД, или в виде файлов на диске, или даже, возможно, в оперативной памяти – этот вопрос лежит за пределами паттерна MVC). В применении к вебу запрос и ответ ходят по HTTP, поэтому можно условно считать, что на этом рисунке пунктиром обозначены заголовки HTTP-запроса и ответа, а сплошными линиями – их тела[6].

Получив Запрос 1, Контроллер его анализирует, и в зависимости от результатов обработки может выдать следующие варианты ответа (почему ответ имеет номер 4, станет понятно из следующих рисунков):

1. Сразу выдать ответ об ошибке (например, при запросе несуществующей страницы отдать только HTTP-заголовок «404 Not found»);

2. Если поступивший Запрос 1 признан корректным, то, в зависимости от того, является он запросом на просмотр или на модификацию данных, Контроллер вызывает соответствующий метод Модели, такой как Save или Load. Данная ситуация рассмотрена на рисунке 2.2.

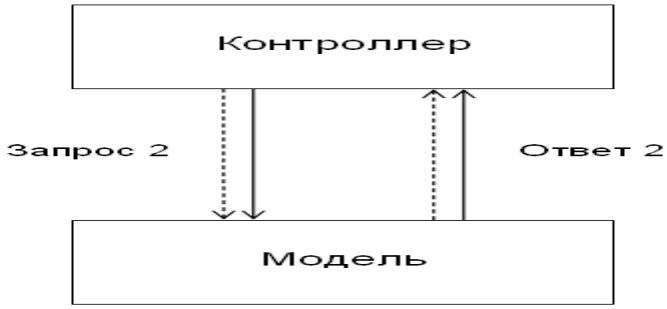

Рис. 2.2. Схема приложения с использованием MVC при наличии двух запросов

Итак, в зависимости от полученного от Модели Ответа 2 Контроллер решает, какое из Представлений вызвать для формирования итогового ответа на изначальный Запрос 1:

1. В случае неудачи – Представление для сообщения об ошибке

2. В случае успеха – Представление для отображения запрашиваемых данных либо сообщения об их успешном сохранении (если Запрос 1 был на изменение данных). Схема подобной ситуации представлена на рисунке 2.3

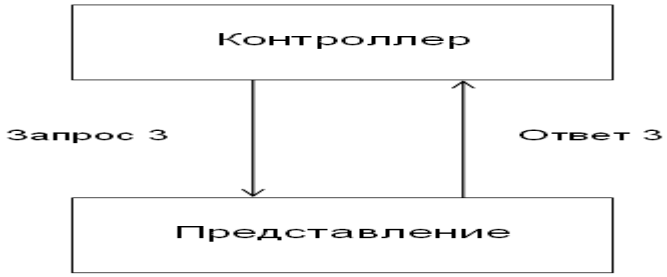

Рис. 2.3. Отображение запрашиваемых данных

Вопрос о том, кто должен проверять на валидность и права доступа входные данные (Контроллер или Модель), является предметом достаточно многочисленных споров, поскольку паттерн MVC не описывает таких деталей.

Построение приложением с использованием шаблона MVC в фреймворке Yii имеет определенные особенности,он использует фронт-контроллер, называемый приложением (application), который инкапсулирует контекст обработки запроса. Приложение собирает информацию о запросе и передает её для дальнейшей обработки соответствующему контроллеру[6].

На рисунке 2.4 рассматривается диаграмма, которая описывает типичную последовательность процесса обработки пользовательского запроса приложением.

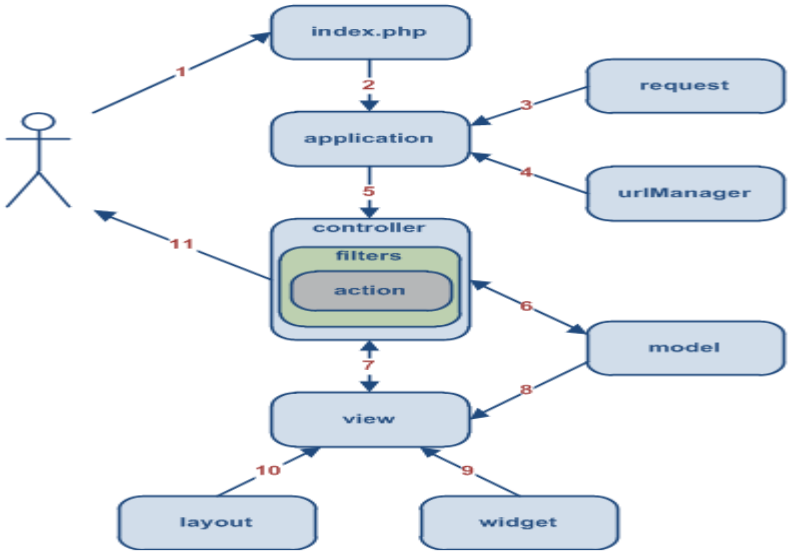

Рис. 2.4. Типичная последовательность работы приложения Yii

Этапы, представленные на диаграмме можно описать как:

1. Пользователь осуществляет запрос посредством URL http://www.example.com/index.php?r=post/show&id=1, и веб-сервер обрабатывает его, запуская скрипт инициализации index.php.
2. Скрипт инициализации создает экземпляр приложения и запускает его на выполнение.
3. Приложение получает подробную информацию о запросе пользователя от компонента приложения request.

4. Приложение определяет запрошенные контроллер и действие при помощи компонента urlManager. В данном примере контроллером будет post, относящийся к классу PostController, а действием — show, суть которого определяется контроллером.
5. Приложение создаёт экземпляр запрашиваемого контроллера для дальнейшей обработки запроса пользователя. Контроллер определяет соответствие действия show методу actionShow в классе контроллера. Далее создаются и применяются фильтры (например, access control, benchmarking), связанные с данным действием, и, если фильтры позволяют, действие выполняется.
6. Действие считывает из базы данных модель Post с ID равным 1.
7. Действие подключает представление show, передавая в него модель Post.
8. Представление получает и отображает атрибуты модели Post.
9. Представление подключает некоторые виджеты.
10. Сформированное представление вставляется в макет страницы.
11. Действие завершает формирование представления и выводит результат пользователю.

2.3 Структура базы данных

С целью хранения, добавления и изменения информации – требуется разработать структуру базы данных, которая в дальнейшем будет реализована.

Главной задачей при построение базы данных является ее нормализация, под понятием нормализации следует принимать процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости. Избыточность данных приводит к непродуктивному расходованию свободного места на диске и затрудняет

обслуживание баз данных. Например, если данные, хранящиеся в нескольких местах, потребуется изменить, в них придется внести одни и те же изменения во всех этих местах. Существует несколько правил нормализации баз данных. Каждое правило называется «нормальной формой». Если выполняется первое правило, говорят, что база данных представлена в «первой нормальной форме». Если выполняются три первых правила, считается, что база данных представлена в «третьей нормальной форме». Есть и другие уровни нормализации, однако для большинства приложений достаточно нормализовать базы данных до третьей нормальной формы [7].

Первая нормальная форма:

- Устраните повторяющиеся группы в отдельных таблицах.
- Создайте отдельную таблицу для каждого набора связанных данных.
- Идентифицируйте каждый набор связанных данных с помощью первичного ключа.

Не используйте несколько полей в одной таблице для хранения похожих данных. Например, для слежения за товаром, который закупается у двух разных поставщиков, можно создать запись с полями, определяющими код первого поставщика и код второго поставщика. Что произойдет при добавлении третьего поставщика? Добавление третьего поля нежелательно, так как для этого нужно изменять программу и таблицу, поэтому данный способ плохо адаптируется к динамическому изменению числа поставщиков. Вместо этого можно поместить все сведения о поставщиках в отдельную таблицу Vendors (поставщики) и связать товары с поставщиками с помощью кодов товаров или поставщиков с товарами с помощью кодов поставщиков.

Вторая нормальная форма:

- Создайте отдельные таблицы для наборов значений, относящихся к нескольким записям.
- Свяжите эти таблицы с помощью внешнего ключа.

Записи могут зависеть только от первичного ключа таблицы (составного ключа, если необходимо). Возьмем для примера адрес клиента в системе

бухгалтерского учета. Этот адрес необходим не только таблице Customers, но и таблицам Orders, Shipping, Invoices, Accounts Receivable и Collections. Вместо того чтобы хранить адрес клиента как отдельный элемент в каждой из этих таблиц, храните его в одном месте: или в таблице Customers, или в отдельной таблице Addresses.

Третья нормальная форма:

- Устраните поля, не зависящие от ключа.

Значения, входящие в запись и не являющиеся частью ключа этой записи, не принадлежат таблице. Если содержимое группы полей может относиться более чем к одной записи в таблице, подумайте о том, не поместить ли эти поля в отдельную таблицу.

Например, в таблицу Employee Recruitment (наем сотрудников) можно включить адрес кандидата и название университета, в котором он получил образование. Однако для организации групповой почтовой рассылки необходим полный список университетов. Если сведения об университетах будут храниться в таблице Candidates, составить список университетов при отсутствии кандидатов не получится. Таким образом, создайте вместо этого отдельную таблицу Universities и свяжите ее с таблицей Candidates при помощи ключа — кода университета.

Выполнять нормализацию базы данных до третьей нормальной формы теоретически желательно, но не всегда практично. Например, для устранения всех возможных зависимостей между полями таблицы Customers придется создать отдельные таблицы для хранения сведений о городах, почтовых индексах, торговых представителях, категориях клиентов и любых других сведений, которые могут дублироваться в нескольких записях. С теоретической точки зрения нормализация желательна. Однако значительное увеличение числа маленьких таблиц может привести к снижению производительности СУБД или исчерпанию памяти и числа дескрипторов открытых файлов [8].

Выполнять нормализацию до третьей нормальной формы может быть целесообразно только для часто изменяемых данных. Если при этом сохранятся зависимые поля, спроектируйте приложение так, чтобы при изменении одного из этих полей пользователь должен был проверить все связанные поля.

Общий вид базы данных, которая получилась в процессе проектирования отображается с помощью логической модели, которая была построена в Erwin Data Modeler. Логическая модель позволяет рассмотреть типы связи между таблицами, название таблиц и полей написаны на русском языке (см. рис. 2.5).

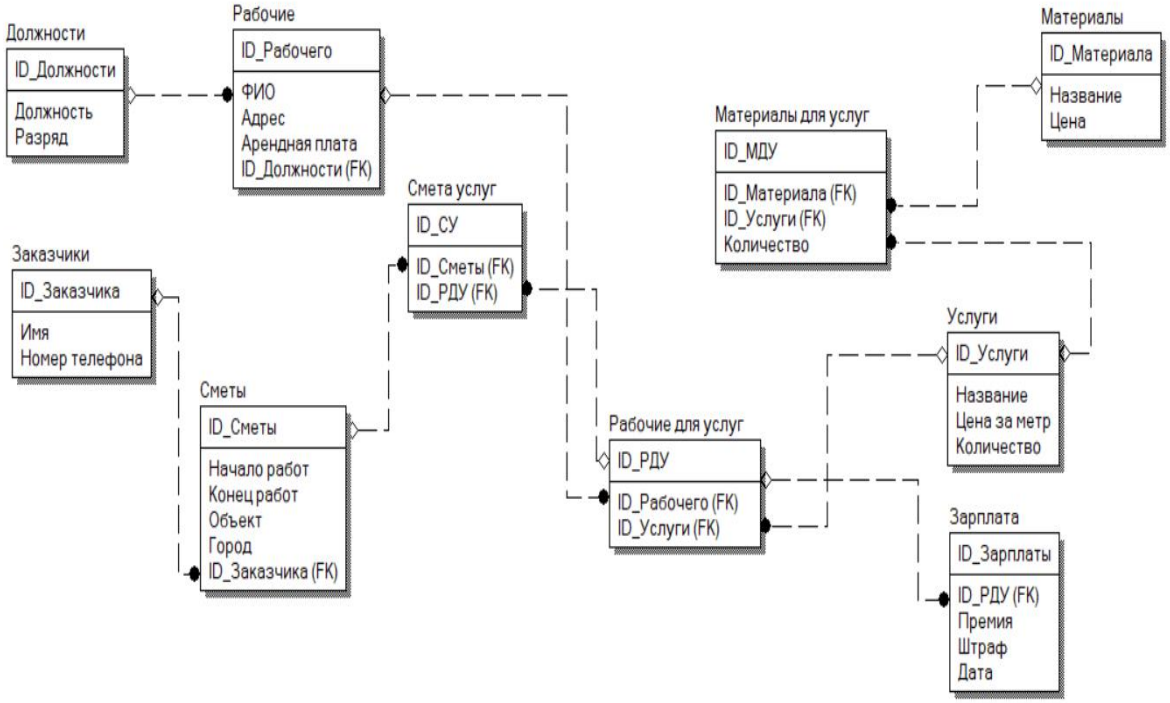

Рис. 2.5. Логическая модель базы данных

В качестве визуального редактора базы данных был выбран онлайн ресурс по адресу: https://www.dbdesigner.net.

Для подробного описания спроектированной базы, будет представлена ее физическая модель на рисунке 2.6, затем каждая таблица будет рассмотрена

отдельно. В первую очередь, рассматриваются таблицы, в которых отсутствуют внешние ключи.

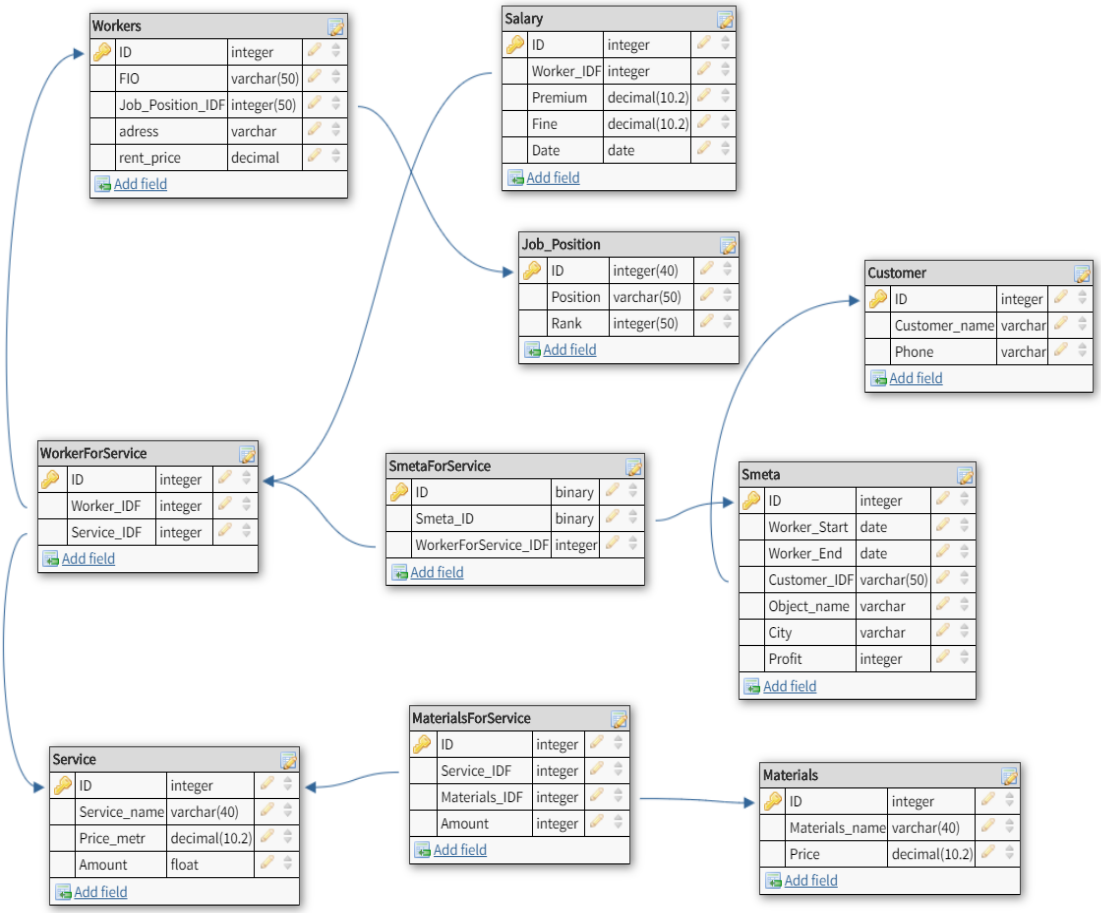

Рис. 2.6. Физическая модель базы данных

.

Таблица «Service» спроектирована с целью хранения в себе списка услуг, которые может оказывать организация, и имеет в себе 4 колонки:

1) service_id - первичный ключ, имеет тип данных int;

2) service_name – название услуги, имеет тип данных varchar;

3) price_metr – цена за метр работы, имеет тип данных numeric;

4) amount – объем выполненной работы, имеет тип данных double.

Таблица «Materials» спроектирована с целью хранения в себе списка расходуемых материалов, которые необходимы для выполнения услуг, и имеет в себе 3 колонки:

1) materials_id - первичный ключ, имеет тип данных int;

2) materials_name – название расходуемого материала, имеет тип данных varchar;

3) price – цена за расходуемый материал, имеет тип данных numeric.

Таблица «Materials_For_Service» спроектирована с целью хранения в себе материалов, необходимых для определенной услуги и таким образом, что дает возможность содержать для каждой услуги неограниченное количество материалов для его выполнения, и имеет в себе 4 колонки:

1) m_f_s_id - первичный ключ, имеет тип данных int;

2) service_id_f – внешний ключ, созданный для связывания с таблицей service, имеет тип данных int;

3) materials_id_f – внешний ключ, созданный для связывания с таблицей materials, имеет тип данных int;

Таблица «Workers_For_Service» спроектирована с целью хранения в себе информации о том, какие работники выполняют какую работу, чтобы определить затем сумму их сдельной заработной платы и имеет в себе 3 колонки:

1) w_f_s_id - первичный ключ, имеет тип данных int;

2) worker_id_f – внешний ключ, созданный для связывания с таблицей workers, имеет тип данных int;

3) service_id_f – внешний ключ, созданный для связывания с таблицей service, имеет тип данных int;

Таблица «Salary» спроектирована с целью хранения в себе информации о заработной платы сотрудникам, сама сумма рассчитывается в приложении и выводится отдельно и имеет 5 колонок:

1) salary_id – первичный ключ, имеет тип данных int;

2) worker_id_f – внешний ключ, созданный для связывания с таблицей workers, имеет тип данных int;

3) premium – колонка необходимая для хранения информации о премии сотрудникам, имеет тип данных numeric;

4) fine – колонка необходимая для хранения информации о штрафах сотрудникам, имеет тип данных numeric;

5) date – поле, которое содержит в себе данные о дате начисления заработной платы.

Таблица «Smeta» спроектирована для хранения в себе списка смет, согласно которым проводятся работы из таблицы «Service» и имеет 7 колонок:

1) smeta_id – первичный ключ, имеет тип данных int;

2) worker_start – колонка, необходимая для определения даты начала работ по смете и имеет тип date;

3) worker_end – колонка, необходимая для определения даты конца работ по смете и имеет тип date;

4) customer_id_f – внешний ключ, который связывает между собой таблицу смет и список заказчиков по смете, имеет тип int;

5) object_name – столбец таблицы, в котором находится информация о объекте, на котором будут производится работы согласно смете, имеет тип varchar;

6) city – столбец таблицы, где будет хранится информация о городе, где будут производится работы, имеет тип varchar;

7) profit – в данной колонке будет храниться процентная прибыль по смете, для того, чтобы определить успешность выполнения работ, имеет тип double.

Таблица «Smeta For Service» спроектирована с целью объединения таблиц смет и оказываемых услуг с целью того, чтобы можно было оказывать несколько услуг по каждой смете и определять рабочих, которые будут работать по определенной смете

1) smeta_for service_id – первичный ключ, имеет тип данных int;

2) smeta_id_f – поле таблицы, содержащее в себе внешний ключ, необходимый для связи с таблицей «Smeta», имеет тип int

3) wfs_id_f – поле таблицы, содержащее в себе внешний ключ, необходимый для связи с таблицей «Worker_For_Service», имеет тип int

Путем анализа спроектированной базы данных можно сделать вывод, что она полностью соответствует третьей форме нормализации. Алгоритм работы с базой данных заключается в том, что необходимо получить расчет заработной платы, и полные расходы на предприятии, для этого заложены в таблице «Service» соответствующие услуги, после этого для каждой услуги через связующую таблицу «Materials For Service» соотносятся необходимые расходуемые материалы из таблицы «Materials», для определения разницы сумм между ценами на услуги и расходуемые материалы на них.

Следующим этапом является определение того, какие услуги выполняет каждый рабочий, для этого заполняются соответствующие поля в связующей таблице «Worker For Service». Полученные первичные ключи из этой таблицы необходимы для того, чтобы связать рабочего и его услугу с оформленным договором из таблицы «Smeta», результирующие значение будет находится в таблице «Smeta For Service». Отталкиваясь от этой таблицы мы можем подсчитать сметную прибыль, умножив разницу между ценой на услуги и материалами для них на определенный коэффициент заложенный в полях таблицы, а также разделить сумму по сметной прибыли между работниками, которая составляет 15 процентов от всей суммы. Сумма, разделенная между рабочими от каждой сметы – сдельная зарплата, результирующая сумма получается путем сложения денежных масс от выполнения услуг с учетом премий и вычетов в следствии штрафов. С учетом того, что из каждой сделки вычитывается заработная плата, то общий доход определится суммой, оставшейся после этих математических операций денежных средств.

2.4 Описание разработанного приложения

Итогом поставленных целей и задач, после проектировки базы данных, является готовое web – приложение. На данном участке работы, мы предоставим для анализа фрагменты кода основных модулей автоматизированной системы

Так как приложение представляет собой административную панель и спроектировано согласно технологии MVC, с использованием фреймворка YII2, код будет максимально укорочен и прост для анализа. Для вывода, изменения и удаления данных, был использован виджет GRIDVIEW, встроенный в фреймворк [6]. Для начала, мы разберем участок кода, где производится вывод данных на страницу (см. листинг 2.1)

Листинг 2.1. Вывод данных из таблицы «Workers»

```
public function actionIndex()
{
$searchModel = new WorkerSearch();
$model = new Worker();
$model->load(Yii::$app->request->post()) && $model->save();
```

Конец листинга

Данный листинг разберем по порядку, контроллер создает объект модели Worker, занося в него все данные из базы данных, которые хранятся в модели, помимо этого в модели содержатся правила, благодаря которым, пользователь не сможет добавлять через приложения записи в таблицу, которые не соответствуют типу данных заложенных в БД. После этого, контроллер генерирует страницу, где возвращаемым значением в представление является массив, благодаря которому можно в представлении вывести соответствующую колонку таблицы [9]. В представлении происходит вызов объекта модели Workers, из контроллера и с помощью виджета Gridview происходит вывод данных в виде таблицы. Так же в этом же контроллере производится добавление записей в таблицу (см. листинг 2.2).

Используя встроенный в фреймворк виджет ActiveRecord, создается специальная форма, которая содержит в каждом из своих текстовых полей – столбец таблицы Workers, после того, как пользователь введет данных и нажимает кнопку «Отправить», массив значений отправляется методом POST

на контроллер [12]. В функции контроллера, код которого можно было увидеть на листинге 2.1 производится проверка на то, какие данные пришли и пришли ли они вообще, если полученные значения подходят по правилам, указанным в модели Worker, происходит операция типа INSERT [11].

Листинг 2.2. Добавление записи в таблицу Workers

```
<?php $form = ActiveForm::begin(['id'=>$model->formName()]); ?>
<?= $form->field($model, 'full_name')->textInput() ?>
<?php echo $form->field($model,'job_position_id_f')-
>dropDownList(app\models\JobsPosition::getPositionList()) ?>
<?= $form->field($model, 'adress')->textInput() ?>
<?= $form->field($model, 'rent_price')->textInput() ?>
<?= Html::submitButton($model->isNewRecord ? 'Добавить' : 'Update', ['class'
=> $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
```

Конец листинга

Процесс изменения записей аналогичен, единственное отличие заключается в том, что запрос имеет тип UPDATE и в запрос передается id записи, которую нужно изменить (см. листинг 2.3).

Листинг 2.3. Изменение записи в таблице Workers

```
$this->registerJs("
$('.grid-view tbody tr').on('click',function(){
var data=$(this).data();
$('#modal-info').modal('show');
$('#modal-info').find('.modal-body').load('/worker/update?id='+data.key);
```

Конец листинга

Алгоритм заключается в том, что с помощью технологии AJAX и языка программирования JS, мы устанавливаем, чтобы при нажатии на

определенную строку таблицы, которую необходимо изменить, происходит вызов модального окна с формой на изменение. Благодаря JS мы можем вытянуть значение переменной data.key, которая равняется id записи, эта полученная переменная передается в запрос и осуществляется изменение записи[9].

Чтобы осуществить возможность производить экспорт в excel, pdf или просто на печать, был установлен соответствующий виджет Exportmenu, который работает параллельно с виджетом GridView (см. листинг 2.5).

Листинг 2.5. Виджет Exportmenu для экспорта данных из таблицы

```
$gridColumns=[ 'full_name',
['attribute'=>'job_position_id_f','value'=>function($model){return
$model>jobPositionIdF->position;}],'adress','rent_price'];
ExportMenu::widget([dataProvider'=>$dataProvider,
'columns'=>$gridColumns])
```

Конец листинга

Мы привязываем виджет к GridView и указываем какие поля из таблиц нам необходимо экспортировать, и указываем место для вывода кнопки, благодаря которой мы будем работать с данным функционалом.

Одним из основополагающих функционалов приложения считается подсчет расходов на предприятии и отталкиваясь от этого оценить рентабельность всех проведенных работ (см. листинг 2.6).

Листинг 2.6. Подсчет расходов на предприятии

```
foreach ($materials as $models){
$materials_col+=$models->amount*$models->materialsIdF->price;}
foreach ($service as $models) {
$result += $models->price_metr *$models->yardage;}
```

Конец листинга

Расчет происходит на основе того, какие услуги выполнялись в организации, внесенные в сметы, данные значения получаются из таблицы «Оказываемые Услуги», поле с ценой на метр работ умножается на объем и из этого значения вычитается сумма, потраченная на расходуемые материалы из таблицы «Управление расходуемыми материалами» и заработную плату сотрудников, которая будет высчитываться согласно сдельной выработке (см. листинг 2.7).

Листинг 2.7. Подсчет сдельной зарплаты сотрудникам

```
['attribute'=>'service_id_f','value'=>function($model){return $model->serviceIdF->service_name;},'filter'=>app\models\Service::getServiceList()], 'amount',
['attribute'=>'Pay','label'=>'Цена','value'=>function($model){return $result=($model->amount*$model->materialsIdF->price) ;}
```

Конец листинга

Сумма формируется отталкиваясь от того, на какую сумму была произведена услуга из таблицы «Service» конкретным сотрудником, расходы на материалы не учитываются, помимо этого, каждый рабочий может получить премию или штраф, итоговая сумма получается без вычета налогов, после чего ведомость экспортируется в excel и передается в бухгалтерию [10].

2.5 Разработка графического интерфейса

Приложение должно предусматривать в себе удобное управление и определенный графический интерфейс, который позволяет отображать данные и иметь функционал управления отображаемых данных.

На начальном этапе построения дизайна клиент – серверного приложения – форма авторизации пользователей (см. рис.2.7).

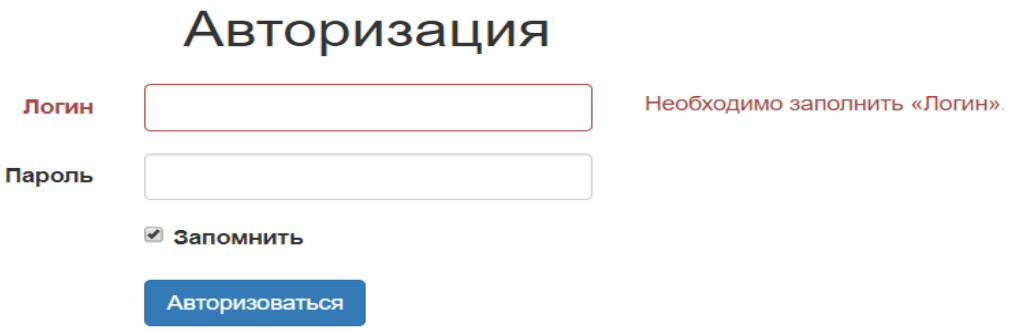

Рис. 2.7. Форма авторизации.

Форма состоит из двух text полей для логина и пароля, checkbutton поля для запоминания отправленных данных и submit кнопки, которая отправляет данные на сервер. Реализовано ограничение по длине введенных символов, и проверка на пустоту полей [13].

Для авторизованного пользователя разработан интерфейс, который за счет своей простоты позволяет быстро освоить основной функционал (см. рис. 2.8).

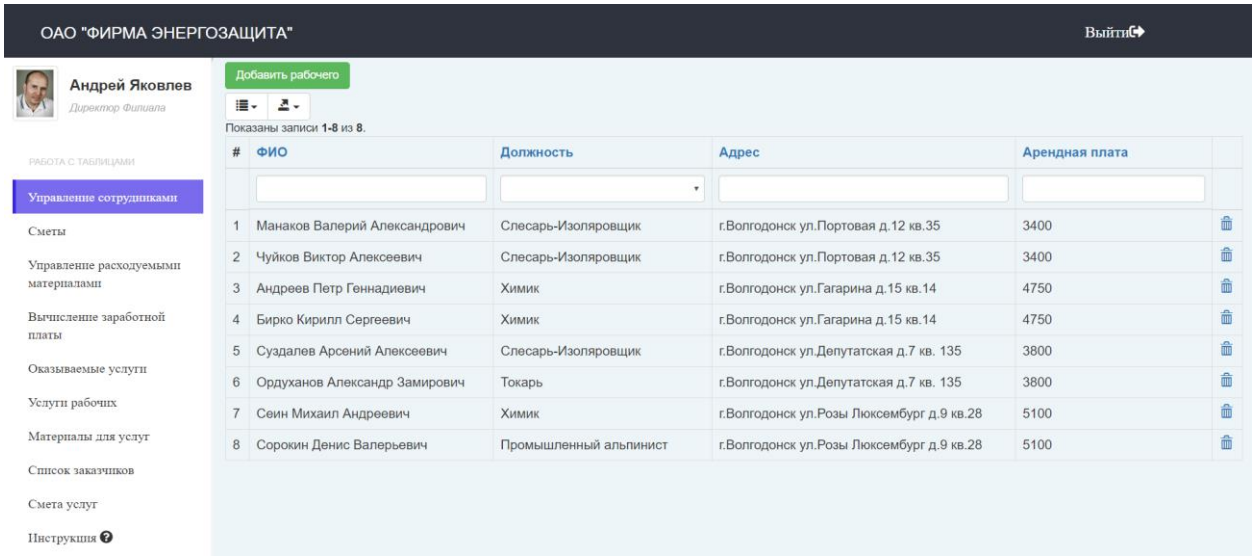

Рис. 2.8. Графическое оформление приложения.

Для навигации в авторизованной системе предусмотрено вертикальное меню, которое представляет в себе определенный набор ссылок на вкладки, где происходит взаимодействие с таблицами. При наведении на каждую из

ссылок происходит подсветка. Кнопка – ссылка для выхода из приложения находится в горизонтальном меню. Отображения данных происходит в виде таблицы, где предусмотрено форматирование по столбцам и строкам.

Заполнение таблиц происходит из формы, вызов которой осуществляется путем нажатия определенной кнопки, которая является частью виджета Modal, который предустановлен фреймворком YII (см. рис. 2.9).

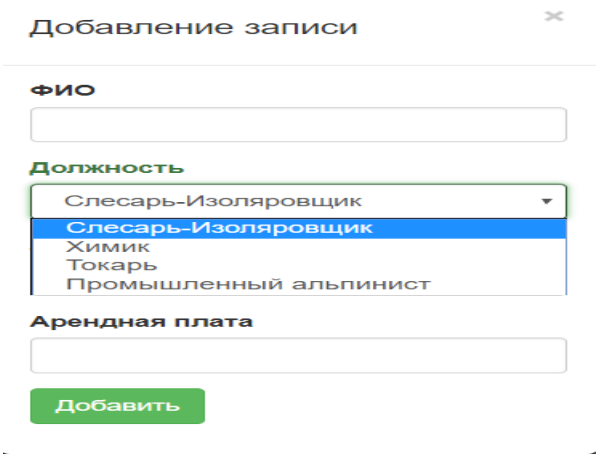

Рис. 2.9. Форма добавления записей

Для заполнения данных о фамилии, адреса и суммы арендной платы – были созданы text поля, должность получается путем обращения к таблице «Управление должностями» и для удобства отображения и занесения информации был добавлен компонент dropDownList, который представляется в виде выпадающего списка, отправляет данные кнопка типа submit.

Аналогично, для записи определенных типов данных, в частности – даты, чтобы сотрудник не имел осложнений в добавлении записей и не думал над правильностью формата заполнения, был добавлен виджет DatePicker (см. рис. 2.10).

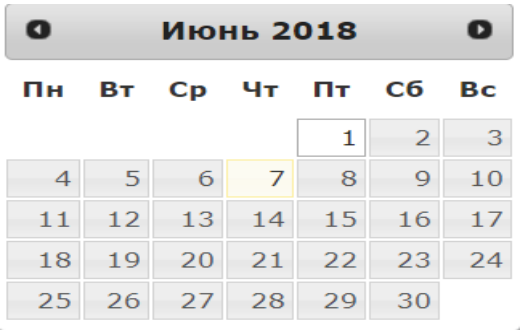

Рис. 2.10. Форма добавления записей

Интерфейс изменения предоставляется аналогично добавлению и происходит в форме, скрытой в модальном окне, вызов происходит путем использования Onclick события в языке JS, с аналогичными полями формы и компонентом dropDownList. Отличие заключается только в том, что для удобства пользователя, в каждое поле формы заранее записаны текущие данные (см. рис.2.11).

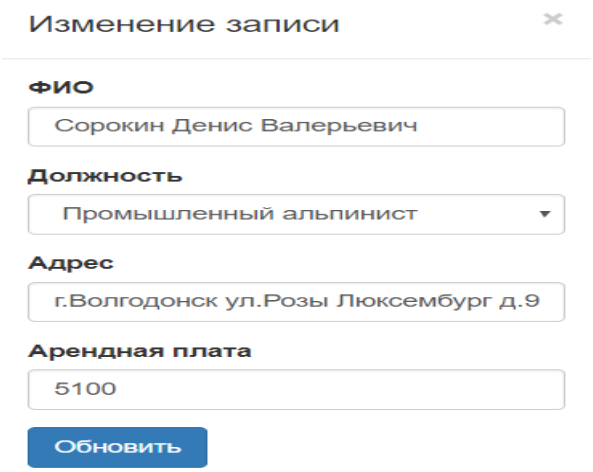

Рис. 2.11. Форма изменения записей

Над таблицей с записями, реализованы поля типа text, которые позволяют сортировать данные после записи ключевого слова или первой буквы, однако, если поле таблицы является отображением записи по внешнему ключу, поиск происходит с помощью dropDownList компонента и соответствующего нему выпадающего списка с записями по внешним ключам.

Глава 3. ТЕСТИРОВАНИЕ И АПРОБАЦИЯ ПРИЛОЖЕНИЯ

3.1 Программа и методика испытаний

В качестве испытуемого объекта будет рассматриваться спроектированное и разработанное ранее приложение «Автоматизированная система расчета расходов материалов и затрат для филиала «Курск энергозащита». Реализованная система имеет административный тип применения, что подразумевает в себе определенного рода функционал.

Целью испытания приложения является проверка работоспособности определенного функционала и соответствие к требованиям, которые были описаны в первой главе. Необходимо выделить основные возможности приложения, которые имеют обязательный характер проверки на соответствие с поставленными задачами:

-возможность вести учет сотрудников, выполняющих работы;

-возможность составления списка оказываемых услуг и расходуемых материалов для реализации определенной услуги;

-возможность авторизоваться в приложении для пресечения неправомерного вмешательства третьих лиц;

-возможность составлять сметы согласно оказываемым услугам и определять рабочих, которые будут работать согласно этой смете;

-возможность осуществлять автоматический расчет финансовой составляющей: стоимость выполняемых работ, расходуемых материалов;

-возможность автоматизированного определения сдельной заработной платы сотрудникам;

-возможность добавления, изменения и удаления записей в каждую таблицу без внедрения сотрудников в программный код приложения;

-возможность сортировки данных по алфавиту или поиска записей по названию;

-возможность экспортирования данных из таблиц в различные типы файлов:excel, pdf.

Отталкиваясь от требований, описанных раннее, был разработан определенный алгоритм испытаний приложения, состоящий из следующих этапов:

Проверка возможности авторизации в приложении:

1. Через браузер зайти на веб - адрес приложения.

2. Заполнить поля формы авторизации, согласно паролю, который выдал администратор и нажать кнопку «Авторизоваться».

3. Если данные введены верно, убедится в том, что произойдет переадресация на административную панель и, в правом верхнем углу, определить – правильно ли заполнены административные данные о вас.

Определения функционала авторизированного пользователя:

1. Выбрать в вертикальном навигационном меню таблицу «Управление сотрудниками».

2. Убедится в наличии списка сотрудников в виде таблицы.

3. Проверить возможность добавления записей в таблицу, для этого необходимо нажать на кнопку «Добавить сотрудника».

4. Произвести заполнение всех полей в появившейся форме, которая находится в сплывающем модальном окне, после этого осуществить отправление формы.

5. Убедиться в том, что добавление произошло успешно

6. Для проверки функционала изменения записей в таблице – следует выбрать строку, которую необходимо редактировать и щелчком мыши нажать на нее.

7. В появившейся форме изменить определенное поле и отправить форму.

8. Убедиться в изменении данных в таблице.

9. Выбрать строку, которая выступит примером для тестирования возможности удаления записей и нажать кнопку в виде мусорного ведра напротив строки.

10. Подтвердить удаление записи в сплывающем окне.

11. Убедиться в отсутствии записи в обновленной таблице.

12. Чтобы проверить экспортирование записей – нажать на кнопку виджета, где происходит выбор строк для экспортирования.

13. Выбрать определенные строки.

14. Нажать на кнопку с выбором типа файлов, в виде которого произойдет экспортирование и подтвердить действие.

15. Проверить в скачанном файле верность экспортированных записей.

16. В поле для осуществления поиска записать существующее значение.

17. Убедиться в выводе данной записи.

18. Для остальных таблиц приложения произвести действия, описанные в пунктах 2.1-2.17.

19. На странице приложения под названием «Смета Услуг» проанализировать результаты подсчета затрат и сверить подсчеты с помощью калькулятора.

20. Во вкладке «Вычисление заработной платы» проанализировать сумму, которая подсчитана на выплату сотрудниками и сверить расчеты с помощью калькулятора.

21. Для возможности работы персоналу, который не имеет определенных навыков в программировании – реализована помощь, для этого необходимо нажать на вкладку «Инструкция» в вертикальном меню и ознакомиться с информацией о том, как пользоваться приложением.

3.2 Испытание автоматизированной системы

Согласно описанному выше алгоритму, необходимо протестировать автоматизированную систему, начнем с того, как происходит авторизация пользователя.

Переходим по веб – адресу приложения с помощью браузера и после того, как оно загрузиться и появиться окно авторизации, заполняем поля и жмем кнопку «Авторизоваться», данный функционал изображен на рисунке 3.1.

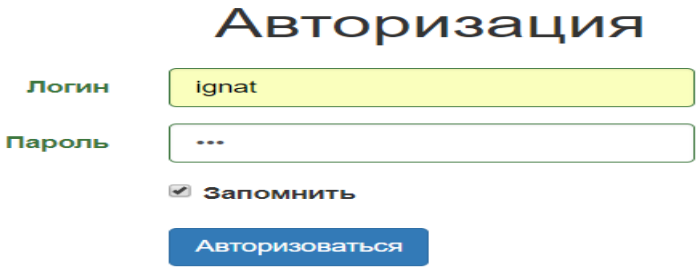

Рис. 3.1 Авторизация пользователя

В результате авторизации происходит переадресация на административную панель приложения, где можно увидеть информацию о авторизованном пользователе (см.рис. 3.2).

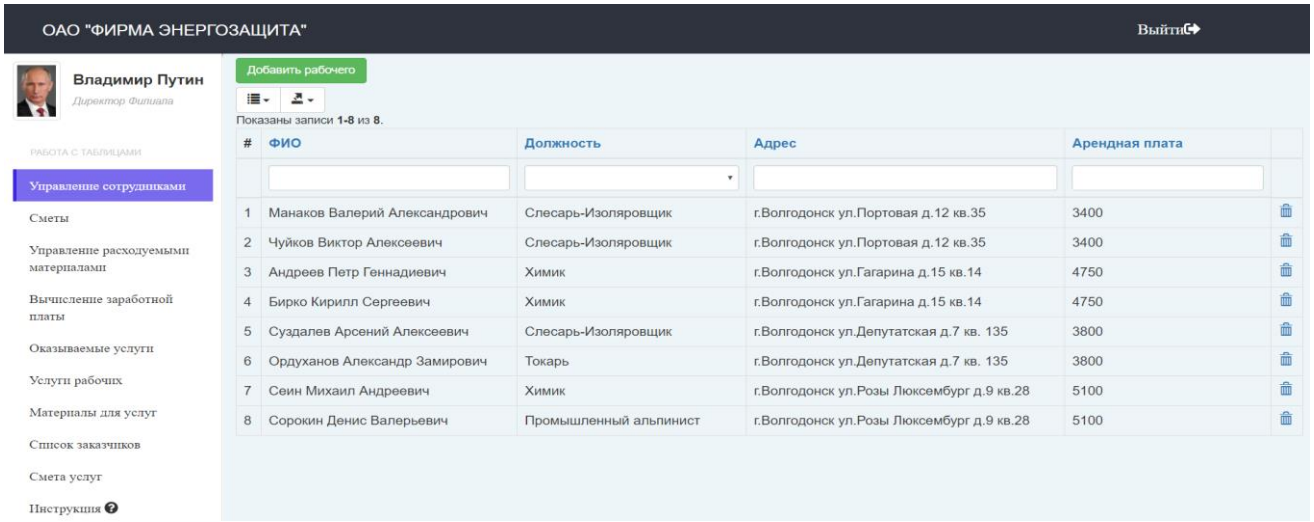

Рис. 3.2. Главная страница с авторизованным пользователем

После того, как пользователь авторизовался и начал пользоваться приложением, необходимо протестировать основной функционал, для этого открываем вкладку «Управление сотрудниками» и нажимаем кнопку – «Добавить рабочего», в появившейся форме, изображенной на рисунке 3.3 заполняем все поля.

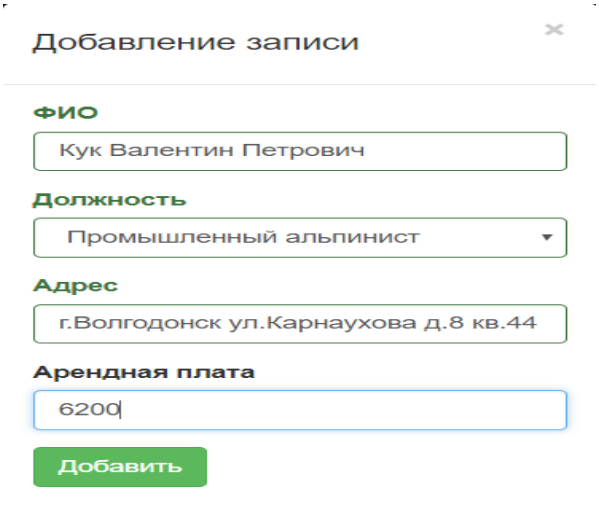

Рис. 3.3. Форма для добавление записей в таблицу «Управление сотрудниками»

После отправления формы, с помощью технологии AJAX происходит асинхронное обновление в таблице и запись добавляется, результат можно увидеть на рисунке 3.4.

#	ФИО	Должность	Адрес	Арендная плата	
1	Манаков Валерий Александрович	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400	
2	Чуйков Виктор Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400	
3	Андреев Петр Геннадиевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750	
4	Бирко Кирилл Сергеевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750	
5	Суздалев Арсений Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Депутатская д.7 кв. 135	3800	
6	Ордуханов Александр Замирович	Токарь	г.Волгодонск ул.Депутатская д.7 кв. 135	3800	
7	Сеин Михаил Андреевич	Химик	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100	
8	Сорокин Денис Валерьевич	Промышленный альпинист	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100	
9	Кук Валентин Петрович	Промышленный альпинист	г.Волгодонск ул.Карнаухова д.8 кв.44	6200	

Рис. 3.4. Обновленная в таблица «Управление сотрудниками»

Можно убедиться, что запись добавлена, на примере этого, необходимо произвести аналогичную проверку на остальные функциональные возможности приложения. Следующим этапом работы с приложением – является редактирование записи, добавленную ниже запись, с сотрудником по фамилии «Кук», мы изменим отчество с «Петрович» на «Иванович». Для этого, мы щелчком мыши выберем определенную строку с данной записью и в появившейся форме поменяем отчество, пример данной операции можно увидеть на рисунке 3.5.

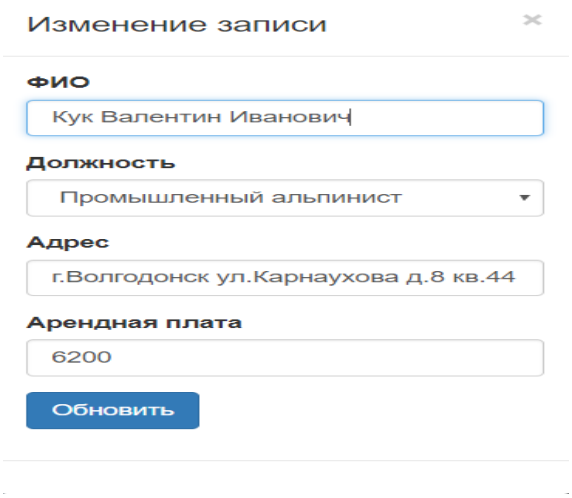

Рис. 3.5. Форма для изменения записей с обновленной фамилией

В результате отправления формы – происходит замена исходных данных на новые и таблица изменяется, результат изменения изображен на рисунке 3.6.

#	ФИО	Должность	Адрес	Арендная плата	
1	Манаков Валерий Александрович	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400	
2	Чуйков Виктор Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400	
3	Андреев Петр Геннадиевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750	
4	Бирко Кирилл Сергеевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750	
5	Суздалев Арсений Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Депутатская д.7 кв. 135	3800	
6	Ордуханов Александр Замирович	Токарь	г.Волгодонск ул.Депутатская д.7 кв. 135	3800	
7	Сеин Михаил Андреевич	Химик	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100	
8	Сорокин Денис Валерьевич	Промышленный альпинист	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100	
9	Кук Валентин Иванович	Промышленный альпинист	г.Волгодонск ул.Карнаухова д.8 кв.44	6200	

Рис. 3.6 Обновленная таблица после редактирование записи

Административная панель позволяет осуществлять и удаление записей из таблицы, для этого необходимо нажать на иконку удаления напротив каждой строки и подтвердить удаление (см. рис. 3.7).

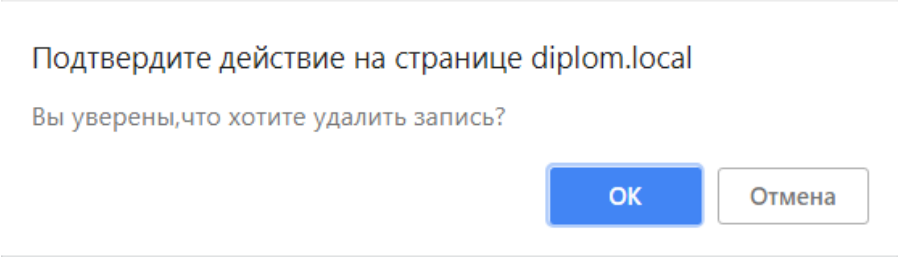

Рис. 3.7. Подтверждение удаления записи

В связи с увеличением записей в базе данных, стало необходимым реализовать возможность сортировки их по алфавиту, возрастанию, убыванию и поиска по базе данным определенных записей. Пример работы поиска по таблице можно увидеть на рисунке 3.8, мы записываем в соответствующее поле определенную фамилию и получаем единственную строку с искомыми данными.

#	ФИО	Должность	Адрес	Арендная плата	
	Сеин				
1	Сеин Михаил Андреевич	Химик	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100	

Рис. 3.8. Результат поиска

Сортировка по возрастанию наиболее актуальна в таблицах с подсчетом расходуемых материалов или оказываемых услуг, чтобы оценить уровень цен на них (см. рис. 3.9).

#	Название материала	Цена	
1	Гвозди 100мм	15.84	
2	Стекловолокно ЭЗ-100	27.00	
3	Вспененный полиэтилен	70.45	
4	Коробка монтажная огнестойкая Гефест КМ-О (4к)-IP41	173.68	
5	Жидкая теплоизоляция металла	367	
6	Державка для Bi-Metall коронок, 32-152 мм	645.00	
7	Пенополистирол ПСБ С 15-О (плотность до 10кг/м3)	2560	

Рис. 3.9. Результат поиска

Результирующая таблица позволяет убедиться в правильности работы поиска по полям таблицы «Управление расходуемыми материалами».

После занесения и обработки данных, в приложении должно быть предусмотрена возможность экспортировать данные из таблиц в формат excel или pdf с дальнейшей печатью. В первую очередь необходим выбрать столбцы таблицы, которые мы будем экспортировать, данный функционал реализуется на рисунке 3.10.

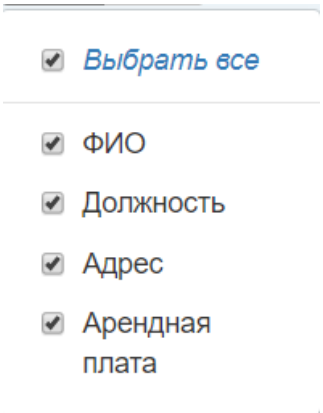

Рис. 3.10. Выбор столбцов для экспортирования

Выбрав данные, мы определяем формат файла, в виде которого произойдет экспортирование и подтвердить операцию (см. рис. 3.11).

Рис. 3.11. Выбор типа файла для экспортирования

В качестве примера, выберем excel 2007+, после подтверждения операции происходит генерирование и скачивание файла, результат проверки качества генерирования excel файла продемонстрирован на рисунке 3.12.

ФИО	Должность	Адрес	Арендная плата
Манаков Валерий Александрович	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400
Чуйков Виктор Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Портовая д.12 кв.35	3400
Андреев Петр Геннадиевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750
Бирко Кирилл Сергеевич	Химик	г.Волгодонск ул.Гагарина д.15 кв.14	4750
Суздалев Арсений Алексеевич	Слесарь-Изоляровщик	г.Волгодонск ул.Депутатская д.7 кв. 135	3800
Ордуханов Александр Замирович	Токарь	г.Волгодонск ул.Депутатская д.7 кв. 135	3800
Сеин Михаил Андреевич	Химик	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100
Сорокин Денис Валерьевич	Промышленный альпинист	г.Волгодонск ул.Розы Люксембург д.9 кв.28	5100

Рис. 3.12. Excel файл после экспортирования данных

Согласно теме работы, наиболее важным функционалом приложения считается расчет расходов, они рассчитываются отталкиваясь от того, сколько произойдет работ сметам, список оказываемых услуг можно увидеть на рисунке 3.13.

#	Название услуги	Цена за метр	Площадь	К оплате	
1	Устройство наливных полов	567.13	210	119097.3	
2	Изоляция трубопроводов матами из супертонкого стекловолокна	370	1500	555000	
3	Изготовление вышек модульных системы "ВМ" и универсальных стоечных лесов системы "УСЛ"	1400	215	301000	
4	Обмуровка	420	470	197400	
5	Антикоррозионная защита	850	450	382500	
6	Покрытие поверхности изоляции трубопроводов: листами алюминиевых сплавов	120	1125	135000	
7	Устройство мягких кровель	672.23	451	303175.73	

Рис. 3.13. Список оказываемых услуг

Материалы, необходимые для этих услуг хранятся в таблице «Управление расходуемыми материалами, с соответствующими ценами на них, данная таблица была предоставлена раннее, связной для них является таблица «Материалы для услуг» где можно указать количество необходимых материалов для определенной услуги (см.рис. 3.14).

#	Материал	Услуга	Количество	Цена	
1	Гвозди 100мм	Изоляция трубопроводов матами из супертонкого стекловолокна	155	2455.2	
2	Стекловолокно ЭЗ-100	Изоляция трубопроводов матами из супертонкого стекловолокна	140	3780	
3	Пенополистирол ПСБ С 15-О (плотность до 10кг/м3)	Устройство мягких кровель	180	460800	
4	Жидкая теплоизоляция металла	Устройство наливных полов	200	73400	
5	Державка для Bi-Metall коронок, 32-152 мм	Антикоррозионная защита	210	135450	
6	Жидкая теплоизоляция металла	Устройство мягких кровель	180	66060	

Рис. 3.14. Список оказываемых услуг и материалов для них

Для того, чтобы начать выполнять работу на определенном объекте, каждую из оказываемых услуг необходимо записать в смету. В смете будут храниться данные о заказчике, объекте и времени проведения работ и существует графа для определения прибыли по смете. В качестве облегчения заполнения акта начале выполнения работ, была создана таблица, которая является шаблоном для последующего использования (см.рис. 3.15)

#	Заказчик	Объект	Город	Прибыль	Начало работы	Конец работы	
1	Ростовская АЭС	Блок №3	Волгодонск	1.21	2017-02-15	2017-03-15	
2	Ростовская АЭС	Химический цех	Волгодонск	1.32	2017-02-15	2017-03-15	
3	Ростовская АЭС	Химический цех	Волгодонск	1.33	2017-08-03	2017-09-11	

Рис. 3.15. Шаблоны смет

Для связи с таблицей шаблонов и оказываемых услуг, и услуг, была создана специальная структура, благодаря которой это в приложении

подсчитывается сумма, которую заработала организация после выполнения всех работ и вычета расходуемых материалов, заработных плат и суммы на аренду квартир (см.рис. 3.16).

#	Услуга	Заказчик	Город	Процентная прибыль	Начало работ	Конец работ
1	Покрытие поверхности изоляции трубопроводов: листами алюминиевых сплавов	Блок №3	Волгодонск	1.21	2017-02-15	2017-03-15
2	Устройство мягких кровель	Химический цех	Волгодонск	1.33	2017-08-03	2017-09-11
3	Изоляция трубопроводов матами из супертонкого стекловолокна	Химический цех	Волгодонск	1.32	2017-02-15	2017-03-15
4	Антикоррозионная защита	Химический цех	Волгодонск	1.32	2017-02-15	2017-03-15

Сумма по всем услугам составляет: **1993173.03** рублей

Для всех оказываемых услуг, сумма на расходуемые материалы составляет: **741945.2** рублей

Сумма, уходящая на заработную плату сотрудникам = **398634.606** рублей

Сумма, с учетом всех отчислений = **852593.224** рублей

Рис. 3.16. Оформленные сметы по услугам и расчет прибыли организации

Специфика работы организации подразумевает в себе отсутствие определенной месячной ставки в качестве заработной платы, предусмотрена только сдельная зарплата, премия и штрафы. Сумма сдельной заработной платы подсчитывается автоматически в приложении (см.рис. 3.17).

#	Сотрудник	Премия	Штраф	Дата	Сдельно	Итого
1	Манаков Валерий Александрович	5200	1100	2017-02-14	13500	17600
2	Чуйков Виктор Алексеевич	6500	0	2017-02-14	13500	20000
3	Андреев Петр Геннадиевич	4100	2050	2017-02-14	55500	57550
4	Бирко Кирилл Сергеевич	5600	850	2017-02-14	38250	43000
5	Сорокин Денис Валерьевич	9950	3450	2017-02-14	30318	36818
6	Ордуханов Александр Замирович	8350	2100	2017-02-14	30318	36568
7	Сеин Михаил Андреевич	7650	950	2017-02-14	38250	44950
8	Суздалев Арсений Алексеевич	7100	500	2017-02-14	55500	62100

Рис. 3.17. Подсчитанная заработная плата сотрудников

Спроектированная автоматизированная система предназначена для сотрудников организации, которые не имеют навыков в программировании.

Решить эту проблему можно созданием инструкции для работы с приложением. Описан основной функционал, который необходим для руководителя: добавление, удаление, изменение, поиск и экспортирование данных. В вертикальном меню необходимо выбрать соответствующий пункт с инструкцией и в всплывающем окне получить информацию, данный функционал описан на рисунке 3.18.

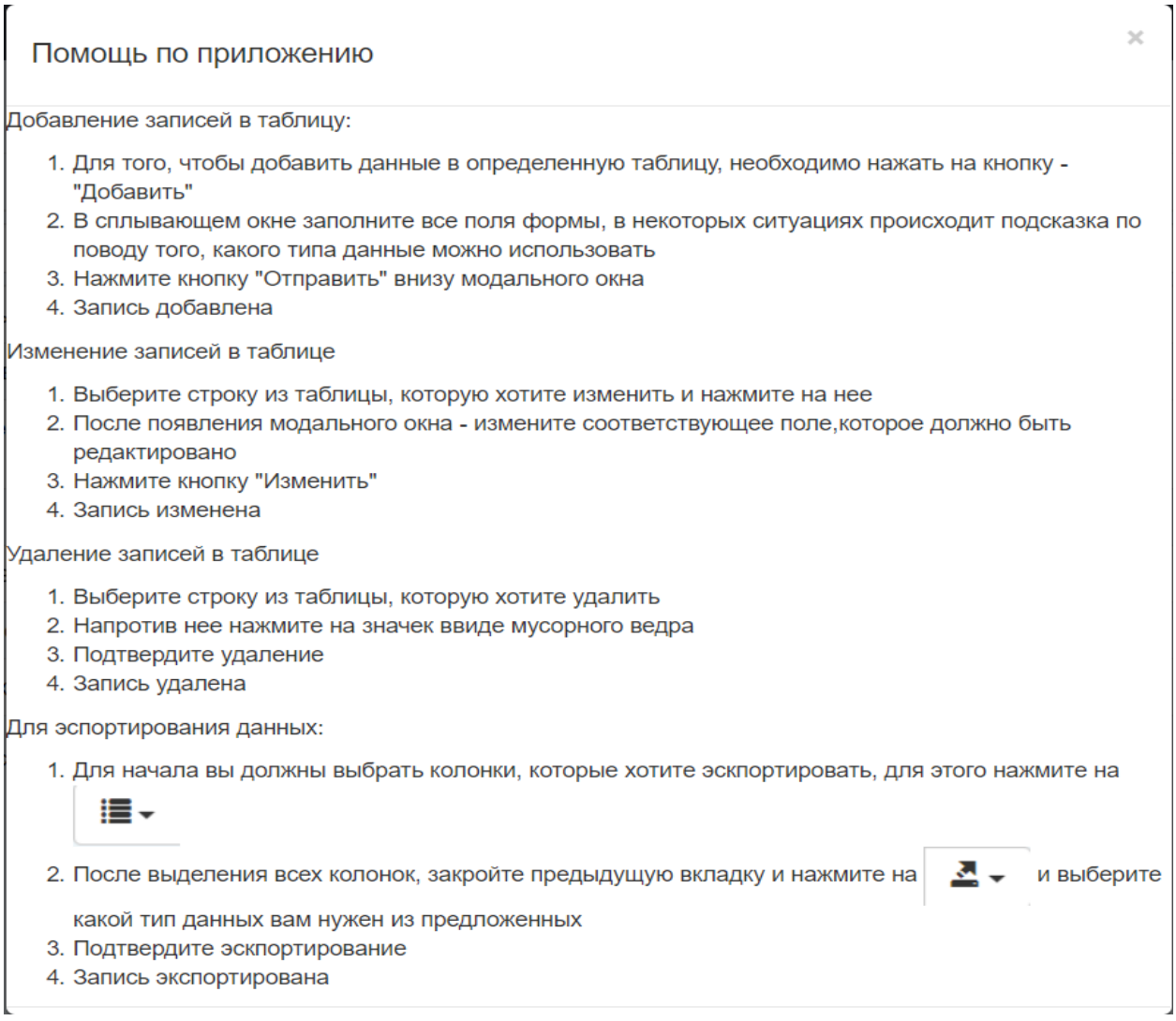

Рис. 3.18. Инструкция по пользованию приложением

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы была спроектирована и разработана автоматизированная система расчета расхода материалов и затрат в ОАО «Фирма Энергозащита», филиал «Курскэнергозащита».

Данная информационно-административная система реализована в форме веб-приложения, и позволяет осуществлять обширный вид управления над существующими данными: добавлять, изменять, удалять, сортировать и экспортировать их. Была предусмотрена возможность авторизации пользователей с защитой от неправомерного вмешательства. Для отправления суммы подсчитанной заработной платы, информации о заработанных средствах путем выполнения заказов согласно смет и расходуемых материалов для этих услуг, был реализован функционал, позволяющих экспортировать данные из таблиц в виде excel файла.

После анализа деятельности клуба, изучения уже имеющихся систем, решающих схожие задачи, были сформулированы требования, на основе которых спроектирована и разработана автоматизированная система. По результатам тестирования системы, следует сделать вывод – разработанное веб приложение полностью удовлетворяет всем требования, предъявленным к системе на этапе постановки задачи. Был составлен акт апробации реализованной системы и в дальнейшем планируется ее внедрение.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1.Преимущества и недостатки 1с: Предприятие- 2017 [Электронный ресурс]. –URL: https://unipro.com.ua (дата обращения: 01.03.2018).

2.Система Галактика - 2017 [Электронный ресурс]. –URL: http://www.erp-online.ru (дата обращения: 03.03.2018).

3.Что такое Парус? - 2017 [Электронный ресурс]. –URL: https://stimul.kiev.ua (дата обращения: 07.03.2018).

4.Что такое YII2, полное руководство по YII - 2016 [Электронный ресурс]. –URL: https://www.yiiframework.com/doc/guide/1.1/ru/quickstart.what-is-yii (дата обращения: 05.04.2018).

5.Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом- 2016 [Электронный ресурс]. –URL: https://habr.com/post/282764/ (дата обращения: 09.04.2018).

6.MVC для веб: проще некуда - 2016 [Электронный ресурс]. –URL: http://savepearlharbor.com/?p=181772 (дата обращения: 12.04.2018).

7.Теоретические основы баз данных - 2015 [Электронный ресурс]. –URL: https://infopedia.su/2x9015.html (дата обращения: 22.04.2018).

8.Описание основных приемов нормализации базы данных- 2015 [Электронный ресурс]. –URL: https://support.microsoft.com/ru-ru/help/283878 (дата обращения: 3.05.2018).

9.Yii2 GridView - виджет таблицы данных - 2017 [Электронный ресурс]. –URL: https://yiico.ru/blog/542-yii2-gridview-vidzhet-tablitsy-dannyh (дата обращения: 11.05.2018).

10.Основы JavaScript - 2017 [Электронный ресурс]. –URL: https://learn.javascript.ru (дата обращения: 13.05.2018).

11.Изучение языка программирования PHP - 2018 [Электронный ресурс]. –URL: https://tproger.ru/tag/php/ (дата обращения: 17.05.2018).

12.Объектно-ориентированный PHP- 2016 [Электронный ресурс]. –URL: https://ruseller.com/lessons.php?id=1145 (дата обращения: 19.05.2018).

13. Продвинутый курс программирования на PHP- 2016 [Электронный ресурс]. –URL: https://php-up.com/ (дата обращения: 22.05.2018).

Приложение 1

Контроллер SalaryController, позволяющий осуществлять подсчет заработной платы представлен ниже.

```
public function actionIndex()
{
$searchModel = new SalarySearch();
$model = new Salary();
$full_zp=Salary::find()->all();
$model->load(Yii::$app->request->post()) && $model->save();
$dataProvider = $searchModel->search(Yii::$app->request->queryParams);
return $this->render('index', [
'searchModel' => $searchModel,
'dataProvider' => $dataProvider,
'model'=>$model,
'full_zp'=>$full_zp
]);
}
public function actionView($id)
{
return $this->render('view', [
'model' => $this->findModel($id),
]);  }
public function actionUpdate($id)
```

```
{
$model = $this->findModel($id);
if ($model->load(Yii::$app->request->post()) && $model->save()) {
return $this->redirect(['index', 'id' => $model->salary_id]);
} else {
if(Yii::$app->request->isAjax)
{
return $this->renderAjax('_form', [
'model' => $model,]);}
else{
return $this->render('update',['model'=>$model]);  }}  }
public function actionDelete($id){
$this->findModel($id)->delete();
if (!Yii::$app->request->isAjax) {
return $this->redirect(['index']);}}
protected function findModel($id){
if (($model = Salary::findOne($id)) !== null) {
return $model;
} else {
throw new NotFoundHttpException('The requested page does not exist.'); }}}
```

Модель Salary, в которой описаны правила работы с базой данных:

```
class Salary extends \yii\db\ActiveRecord
{
```

```
public static function tableName()
{
return 'salary';
}
public function rules()
{
return [
[['premium', 'fine'], 'number'],
[['date'], 'safe'],
[['worker_id_f'], 'integer'],
[['worker_id_f'], 'exist', 'skipOnError' => true, 'targetClass' =>
WorkerForService::className(), 'targetAttribute' => ['worker_id_f' =>
'worker_for_service_id']],];}
public function attributeLabels()
{
return [
'salary_id' => 'Salary ID',
'premium' => 'Премия',
'fine' => 'Штраф',
'date' => 'Дата',
'worker_id_f' => 'Сотрудник',];}
public function getWorkerIdF()
{
```

```
return $this->hasOne(WorkerForService::className(), ['worker_for_service_id'
=> 'worker_id_f']);}}
```

Модель SalarySearch, позволяющая осуществлять поиск и сортировку записей:

```
class SalarySearch extends Salary
{
public function rules(){
return [
[['salary_id', 'worker_id_f'], 'integer'],
[['premium', 'fine'], 'number'],
[['date'], 'safe'],];}
public function scenarios()
{
return Model::scenarios();
}
public function search($params)
{
$query = Salary::find();
$dataProvider = new ActiveDataProvider([
'query' => $query,
]);
$this->load($params);
if (!$this->validate()) {
```

```
return $dataProvider;
}
$query->andFilterWhere([
'salary_id' => $this->salary_id,
'premium' => $this->premium,
'fine' => $this->fine,
'date' => $this->date,
'worker_id_f' => $this->worker_id_f,]);
return $dataProvider;}
```

Представление Index.php, в котором происходит отображение данных, и настройка виджетов:

```
<div class="container-fluid">
<div class="salary-index">
<div class="insert">
<?php
Modal::begin([
'size'=>'modal-sm',
'header' => '<h4>Добавление записи</h4>',
'toggleButton' => ['label' => 'Добавить зарплату',
'tag'=>'button',
'class' => 'btn btn-success'],]);?>
<?php $form = ActiveForm::begin(['id'=>$model->formName()]); ?>
```

```
<?= $form->field($model, 'worker_id_f')->textInput()-
>dropDownList(app\models\WorkerForService::getWorkerForServiceList()) ?>

<?= $form->field($model, 'premium')->textInput() ?>

<?= $form->field($model, 'fine')->textInput() ?>

<?= $form->field($model, 'date')->widget(\yii\jui\DatePicker::class, [

'language' => 'ru',

'dateFormat' => 'yyyy-MM-dd',

]) ?>

<?//= $form->field($model, 'date')->textInput(['type' => 'date']);?>

<div class="form-group">

<?= Html::submitButton($model->isNewRecord ? 'Create' : 'Update', ['class' =>
$model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>

</div>

<?= Html::csrfMetaTags() ?>

<meta charset="<?= Yii::$app->charset ?>">

<title><?= Html::encode($this->title) ?></title>

<?php ActiveForm::end(); ?>

<?php Modal::end(); ?>

</div>

<?php Pjax::begin(['id'=>'salaryGrid', 'enablePushState' => false]); ?>

<?= GridView::widget([

'dataProvider' => $dataProvider,

'filterModel' => $searchModel,
```

```
'columns' => [
['class' => 'yii\grid\SerialColumn'],
['attribute'=>'worker_id_f','value'=>function($model){ return $model->workerIdF->workerIdF->full_name;},'filter'=>app\models\WorkerForService::getWorkerForServiceList()],
'premium','fine','date',
['attribute'=>'Pay','label'=>'Сдельно','value'=>function($model){$result =($model->workerIdF->serviceIdF->price_metr*$model->workerIdF->serviceIdF->yardage)*0.1; return round($result, 0, PHP_ROUND_HALF_UP);}],
['attribute'=>'Pay','label'=>'Итого','value'=>function($model){$result = (($model->workerIdF->serviceIdF->price_metr*$model->workerIdF->serviceIdF->yardage)*0.1)+($model->premium-$model->fine)+405*22;return round($result, 0, PHP_ROUND_HALF_UP);;}],
['class' => 'yii\grid\ActionColumn',
'template'=>'{delete}',
'buttons' => [
'delete' => function ($url) {
return Html::a('<span class="glyphicon glyphicon-trash"></span>', '#', [
'title' => Yii::t('yii', 'Delete'),
'aria-label' => Yii::t('yii', 'Delete'),
'onclick' => "
if (confirm('Вы уверены,что хотите удалить запись?')) {
$.ajax('$url', {
type: 'POST'
```

```
}).done(function(data) {
$.pjax.reload({container: '#salaryGrid'});});}
event.stopPropagation();",]);},],],],]); ?><?php Pjax::end(); ?></div>
</div>
<div class="modal modal-info fade in" id="modal-info" style="display: none;
padding-right: 16px;"><div class="modal-dialog"><div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-dismiss="modal" aria-label="Close">
<span aria-hidden="true">×</span></button>
<h4 class="modal-title">Info Modal</h4></div>
<div class="modal-body"></div>
<div class="modal-footer"></div></div></div></div>
<?php
foreach ($full_zp as $zp)
{
$sum_zp+=(($zp->workerIdF->serviceIdF->price_metr*$zp->workerIdF-
>serviceIdF->yardage)*0.1)+($zp->premium-$zp->fine)+405*22;}?>
<p> Общая сумма= <?= $sum_zp;?></p>
<?php $this->registerJs("
$('.grid-view tbody tr').on('click',function(){
var data=$(this).data();
$('#modal-info').modal('show');
$('#modal-info').find('.modal-title').text('Изменение записи');
```

```
$('#modal-info').find('.modal-body').load('/salary/update?id='+data.key);})");?>
<?php $script=<<< JS
$('form#{$model->formName()}').on('beforeSubmit',function(e) {
var \$form=$(this);
$.post(\$form.attr("action"),\$form.serialize())
.done(function(result){
$(\$form).trigger("reset");
$.pjax.reload({container:'#salaryGrid'});
})
.fail(function() {
console.log("server error");
})
return false;});
JS;
$this->registerJs($script);
?>
```

<?php

Файл Update.php, в котором происходит вызов формы для изменения данных.

```
$this->title = 'Update Salary: ' . $model->salary_id;
$this->params['breadcrumbs'][] = ['label' => 'Salaries', 'url' => ['index']];
$this->params['breadcrumbs'][] = ['label' => $model->salary_id, 'url' => ['view', 'id' => $model->salary_id]];
$this->params['breadcrumbs'][] = 'Update';
```

```
?>

<div class="salary-update">

<h1><?= Html::encode($this->title) ?></h1>

<?= $this->render('_form', [

'model' => $model, ]) ?></div>
```

Представление Smeta_for_service/index.php, в котором происходит подсчет расходов на предприятии:

```
<?php

foreach ($materials as $models)

{

   $materials_col+=$models->amount*$models->materialsIdF->price;

}

foreach ($service as $models) {

   $result += $models->price_metr *$models->yardage;

}

?>

<?php

foreach ($full_zp as $zp)

{

   $sum_zp+=(($zp->workerIdF->serviceIdF->price_metr*$zp->workerIdF-
>serviceIdF->yardage)*0.1)+($zp->premium-$zp->fine)+405*22;

   $sum_app+=$zp->workerIdF->workerIdF->rent_price;

}?>
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*