**Содержание**

[Перечень сокращений, условных обозначений, терминов 9](#_Toc421573982)

[Введение 10](#_Toc421573983)

[1 Техническое задание на создание системы 11](#_Toc421573984)

[1.1 Назначение и цели создания системы 11](#_Toc421573985)

[1.2 Характеристика объекта автоматизации 11](#_Toc421573986)

[1.2.1 Общее описание 12](#_Toc421573987)

[1.2.2 Структура и принципы функционирования 12](#_Toc421573988)

[1.2.3 Существующая информационная система и её недостатки 12](#_Toc421573989)

[1.2.4 Анализ аналогичных разработок 13](#_Toc421573990)

[1.2.5 Актуальность проводимой разработки 13](#_Toc421573991)

[1.3 Общие требования к системе 14](#_Toc421573992)

[1.3.1 Требования к структуре и функционированию системы 14](#_Toc421573993)

[1.3.2 Дополнительные требования 14](#_Toc421573994)

[1.4 Требования к функциям, выполняемым системой 15](#_Toc421573995)

[1.4.1 Авторизация и аутентификация в системе 15](#_Toc421573996)

[1.4.2 Отображение списка домов, состоящих в региональной программе капитального ремонта за определённый период 15](#_Toc421573997)

[1.4.3 Просмотр информации о сотрудниках 15](#_Toc421573998)

[1.4.4 Просмотр оказываемых услуг по содержанию и ремонту жилья, порядке и условиях оказания 16](#_Toc421573999)

[1.4.5 Просмотр планируемых, проведенных работах и их стоимости 16](#_Toc421574000)

[1.4.6 Просмотр информациио тарифах инормативах на услуги 16](#_Toc421574001)

[1.4.7 Просмотр информации о начислениях, перерасчётах, показаниях приборов учёта, сборах, задолженности по дому 17](#_Toc421574002)

[1.5 Требования к видам обеспечения 17](#_Toc421574003)

[1.5.1 Требования к алгоритмическому обеспечению 17](#_Toc421574004)

[1.5.2 Требования к информационному обеспечению 18](#_Toc421574005)

[1.5.3 Требования к программному обеспечению 19](#_Toc421574006)

[1.5.4 Требования к техническому обеспечению 19](#_Toc421574007)

[1.6 Выводы по техническому заданию на создание системы 20](#_Toc421574008)

[2 Модель данных системы 21](#_Toc421574009)

[2.1 Стандарт функционального моделирования IDEF0 21](#_Toc421574010)

[2.2 IDEF0-модель подсистемы "Портал управляющих компаний" для АИС: Объектовый учёт 23](#_Toc421574011)

[3 Информационное обеспечение системы 27](#_Toc421574012)

[3.1 Выбор технологий управления данными 27](#_Toc421574013)

[3.2 Проектирование базы данных 27](#_Toc421574014)

[3.2.1 Логическая и физическая модели данных 27](#_Toc421574015)

[3.2.2 Проектирование реализации 35](#_Toc421574016)

[3.3 Организация сбора, передачи, обработки и выдачи информации 36](#_Toc421574017)

[3.3.1 Сбор информации 36](#_Toc421574018)

[3.3.2 Передача информации 36](#_Toc421574019)

[3.3.3 Обеспечение достоверности 37](#_Toc421574020)

[3.3.4 Процедуры обслуживания 37](#_Toc421574021)

[3.3.5 Выдача информации 37](#_Toc421574022)

[4 Алгоритмическое обеспечение системы 38](#_Toc421574023)

[4.1 Алгоритм автодополнения 38](#_Toc421574024)

[4.2 Алгоритм фильтра по адресу 41](#_Toc421574025)

[5 Программное обеспечение системы 43](#_Toc421574026)

[5.1 Выбор компонентов программного обеспечения 43](#_Toc421574027)

[5.1.1 Утилиты и инструменты проектирования базы данных 44](#_Toc421574028)

[5.1.2 Технологии ORM 44](#_Toc421574029)

[5.1.3 Сравнение ORM-фреймворков LINQ to SQL и Entity Framework 45](#_Toc421574030)

[5.2 Разработка прикладного программного обеспечения 47](#_Toc421574031)

[5.2.1 Структура прикладного программного обеспечения 47](#_Toc421574032)

[5.2.2 Программный модуль «Портал управляющих компаний» 49](#_Toc421574033)

[5.2.3 Программный модуль подсистемы «Объектовый учёт» 50](#_Toc421574034)

[5.3 Инсталляция и особенности работы 52](#_Toc421574035)

[5.4 Работа с основными разделами системы 52](#_Toc421574036)

[6 Тестирование системы 56](#_Toc421574037)

[6.1 Условия и порядок тестирования 56](#_Toc421574038)

[6.2 Исходные данные для контрольных примеров 56](#_Toc421574039)

[6.2.1 Форма авторизации 56](#_Toc421574040)

[6.2.2 Стартовая страница 58](#_Toc421574041)

[6.2.3 Страница капитального ремонта 59](#_Toc421574042)

[6.3 Результаты тестирования 60](#_Toc421574043)

[7 Экономический раздел 61](#_Toc421574044)

[7.1 Расчет трудоемкости разработки модулей 61](#_Toc421574045)

[7.2 Расчет показателя трудоемкости для программного продукта 61](#_Toc421574046)

[7.3 Расчет затрат на материальные ресурсы и сырье 64](#_Toc421574047)

[7.4 Расчет затрат на разработку системы 67](#_Toc421574048)

[7.5 Расчет затрат на оплату труда 67](#_Toc421574049)

[7.6 Расчет отчислений на социальные нужды 68](#_Toc421574050)

[7.7 Расчет амортизационных отчислений 68](#_Toc421574051)

[7.8 Себестоимость проекта 69](#_Toc421574052)

[7.9 Расчет показателей экономической эффективности и ожидаемого годового экономического эффекта от внедрения разработки 69](#_Toc421574053)

[7.10 Расчет плановой прибыли 70](#_Toc421574054)

[7.11 Расчет основных технико-экономических показателей и эффективности использования программного продукта 71](#_Toc421574055)

[8 Безопасность и экологичность проекта 73](#_Toc421574056)

[8.1 Исходные данные 73](#_Toc421574057)

[8.2 Перечень нормативных документов 74](#_Toc421574058)

[8.3 Анализ потенциальных опасностей 76](#_Toc421574059)

[8.3.1 Анализ вредных и опасных производственных факторов 77](#_Toc421574060)

[8.3.2 Анализ воздействия на окружающую среду 85](#_Toc421574061)

[8.3.3 Анализ возможных чрезвычайных ситуаций 85](#_Toc421574062)

[8.4 Мероприятия по охране труда 88](#_Toc421574063)

[8.4.1 Мероприятияпо обеспечению комфортных условий труда 90](#_Toc421574064)

[8.4.2 Мероприятия по защите от опасных и вредных производственных факторов 92](#_Toc421574065)

[8.5 Мероприятия по охране окружающей среды 94](#_Toc421574066)

[8.6 Мероприятия по защите от чрезвычайных ситуаций 94](#_Toc421574067)

[8.7 Расчетная часть 96](#_Toc421574068)

[8.7.1 Расчет освещенности 96](#_Toc421574069)

[8.7.2 Расчет уровня шума 98](#_Toc421574070)

[8.8 Оценка эффективности 99](#_Toc421574071)

[Заключение 100](#_Toc421574072)

[Список использованных источников 101](#_Toc421574073)

[Приложение А (обязательное) Текст программы 102](#_Toc421574074)

# Перечень сокращений, условных обозначений, терминов

P2P – peer to peer

БД – база данных

ИС – информационная система

ПО – программное обеспечение

СУБД – система управления базами данных

ЯВУ – язык высокого уровня

# Введение

Во все времена передача информации между пользователями была основной функцией сети Интернет. Одним из вариантов решения данной задачи являются мессенджеры. Данный тип ПО обрел популярность достаточно давно, но так и не потерял ее, а напротив, даже приобрел. Со временем появляется все больше всевозможных сервисов, со своими особенностями и спецификой. Именно из-за набирающей объемы информации и популяризации месенджеров темой моей выпускной квалификационной работы является автоматизированная система обмена сообщениями.

Также, в последнее время, все острее встает вопрос приватности передачи данных. В обществе, где информация становится главной ценностью, люди все больше и больше хотят защитить информацию от попадания третьим лицам. P2P архитектура в свою очередь, как раз таки позволяет устанавливать соединение напрямую между клиентами, минуя сервер и третьих лиц. Плюс ко всему это обеспечивает легкую расширяемость без серьезных вложений, отказоустойчивость, трудности при намерении заблокировать передачу данных третьей стороной, что и послужило причиной использования технологий пиринговой сети.

Сейчас браузер для пользователя становится некой экосистемой, в которой человеку комфортно. В связи с этим многие компании запускают веб-версии, своих десктопных продуктов, предоставляя возможность пользоваться ПО просто открыв новую вкладку, без необходимости установки. Такое решение оказалось очень популярным, несмотря на более медленное функционирование или отсутствие второстепенных функций. Поэтому было выбрано оформление продукта в качестве веб-приложения, тем более, что это еще и решает проблемы кроссплатформенности, а также внушительных затрат и повышенной сложностьи поддержки в случае разработки нативного клиента для каждой из платформ.

1. Техническое задание на создание системы

В данном разделе приводится техническое задание на создание автоматизированной системы обмена сообщениями на основе технологии пиринговой сети.

* 1. Назначение и цели создания системы

Основными целями разработки системы в рамках выпускной квалификационной работы являются:

1. Осуществление передачи информационных сообщений между пользователями системы минуя посторонних лиц;
2. Предоставление распределенного веб-сервиса для обмена данными;
3. Отказоустойчивость системы.
   1. Характеристика объекта автоматизации

Объектом автоматизации информационной системы является процесс обмена сообщениями между пользователями системы.

* + 1. Общее описание

Распределенный веб-сервис на основе технологии пиринговой сети, обеспечивающий передачу сообщений между пользователями системы.

* + 1. Структура и принципы функционирования

Обеспечение работы P2P мессенджера делится на несколько этапов:

1. Наличие распределенной архитектуры;
2. Регистрация;
3. Авторизация;
4. Отправка сообщений;
5. Присваивание уникального идентификационного ID.

Основываясь на структуре, можно выделить основные функции объекта автоматизации:

1. Передача информационных сообщений;
2. Обеспечение приватности;
3. Отказоустойчивость.
   * 1. Анализ аналогичных разработок

В качестве аналогичных разработок можно взять такие приложения, как Skype, Telegram, Bleep.

Если рассматривать Skype и Telegram, то к отличиям можно отнести клиент-серверную архитектуру вместо P2P. Также в Telegram отсутствуют функции видеостриминга, звонков, видеозвонков. В случае с Bleep принципиальных различий, например, таких как архитектура системы, нет, однако, Bleep является системой с нативным клиентом для каждой из платформ, в следствие чего отсутствует веб-сервис. Также в Bleep отсутствует возможность видеостриминга.

* + 1. Актуальность проводимой разработки

Актуальность разработки обусловлена все растущей популярностью мессенджеров, усиливающимся в последнее время желанием пользователя сохранить приватность беседы, а также миграцией десктопного ПО в сеть Интернет в виде веб-сервисов.

* 1. Общие требования к системе

В данном подразделе описаны общие требования к системе.

* + 1. Требования к структуре и функционированию системы

Разрабатываемая система должна состоять из двух частей, реализующих архитектуру пиринговой сети: клиентской части, использующей P2P соединение и серверной, выдающей клиентам уникальные идентификаторы для осуществления P2P соединения.

В рамках модернизации системы можно рассмотреть расширенный функционал, присущий мессенджерам, и поддержку групповых чат-комнат.

* + 1. Дополнительные требования

Дополнительные требования к системе:

1. В системе должен быть реализован механизм, обеспечивающий конфидециальность передачи информации между пользователями;
2. Система должна обрабатывать исключительные ситуации и корректно отображать сообщения об ошибках;
3. Персонал, работающий с информационной системой, должен обладать навыками работы за компьютером и использования интернет-обозревателя;
4. Система должна быть эргономичной. Графический интерфейс пользователя должен отвечать современным требованиям к оформлению веб-сайтов.
5. Документация по выполненной системе должна быть разработана согласно ГОСТ [4, 5, 6, 7].
   1. Требования к функциям, выполняемым системой

В данном подразделе приводится информация об основных функциях, выполняемых системой.

* + 1. Авторизация и регистрация в системе

Данная функция позволяет авторизоваться зарегистрированному пользователю под своими учетными данными.

Приоритет выполнения задачи - наивысший.

Требования к входным данным: входные данные вводятся пользователем в форму аутентификации.

Требования к выходным данным: запись данных о регистрации в базу данных, предоставление доступа пользователю к функциям системы.

* + 1. Добавление контактов

Пользователь может добавлять контакты в список контактов, используя строку поиска и уникальный идентификатор пользователя.

Требования к входным данным: входные данные вводятся пользователем в форму поиска.

Требования к выходным данным: добавление искомого контакта в список контактов. В случае ненахождения контакта – вывод сообщения об этом пользователю.

* + 1. Удаление контактов

Пользователь может удалять контакты из своего списка контактов, используя интерфейс приложения.

* + 1. Передача информации между пользователями

В информационной системе должен быть реализован механизм передачи информации на основе технологии пиринговой сети, то есть передача сообщений напрямую от пользователя к пользователю.

* + 1. Шифрование

Для осуществления конфидецинциальности передаваемой информации используется сквозное шифрование. Данный тип шифрования обеспечивает максимальный уровень защиты для пользователей, которые серьезно заботятся о конфиденциальности данных.

* + 1. Отказоустойчивость

Реализация отказоустойчивости достигается за счет децентрализованной системы, реализованной по принципу P2P, что подразумевает под собой установление соединения между пользователями напрямую, минуя сервер.

* 1. Требования к видам обеспечения

В данном подразделе приводится требования к видам обеспечения для ИС.

* + 1. Требования к алгоритмическому обеспечению

Необходимо разработать алгоритмы, реализующие следующие задачи:

1. Шифрование канала связи;
2. Поиск пользователя по заданному идентификатору;
3. Передача информации по протоколу WebRTC.

При разработке информационной системы необходимо использовать возможности языка и платформы для реализации алгоритмов. Если такая возможность существует, необходимо использовать библиотеки, разрабатываемые по открытой лицензии. При отсутствии такой возможности необходимо самостоятельно реализовать функциональность.

* + 1. Требования к информационному обеспечению

Для хранения данных требуется использовать систему управления реляционными базами данных.

Требования к реляционной СУБД:

1. Наличие триггеров на изменение и удаление записей в таблицах;
2. Расширенная обработка исключительных ситуаций.

Пользователям запрещается получать данные с помощью языка запросов без использования прикладного программного обеспечения.

При использовании системы необходимо обеспечить шифрование данных.

* + 1. Требования к программному обеспечению

Клиентская часть информационной системы должна корректно отображаться и функционировать на современных браузерах, а также в браузере Google Chrome.

Серверная часть информационной системы должна корректно разворачиваться на платформе Node.js.

Информационная система должна разрабатываться в любых из открытых инструментальных средах разработки [1, 10, 11].

* + 1. Требования к техническому обеспечению

Требования к аппаратному обеспечению серверной части информационной системы:

1. Процессор – Intel Core i3;
2. Оперативное запоминающее устройство – 8Гб;
3. Видеоадаптер – OpenCL совместимый;
4. Накопитель на жёстких магнитных дисках - объёмом не менее 1Тб;
5. Сетевой адаптер с пропускной способностью 100 Мбит/с.

Требования к аппаратному обеспечению клиентской части информационной системы:

1. Процессор – Intel Core i3;
2. Оперативное запоминающее устройство – не менее 512Мб;
3. Видеоадаптер – OpenGL совместимый;
4. Накопитель на жёстких магнитных дисках - объёмом не менее 10Гб;
5. Сетевой адаптер;
6. Монитор – цветной с разрешением не менее 1024х768.
   1. Выводы по техническому заданию на создание системы

Объектом автоматизации информационной системы является процесс передачи информации напрямую от одного пользователя другому. На текущий момент передача информации через мессенджеры является довольно популярной практикой и такой подход все больше заслуживает признание среди пользователей. Тем не менее, направление веб-ориентированных мессенджеров только набирает популярность, является перспективным и актуальным.

Разрабатываемая система должна состоять из двух частей: клиентской и серверной, которые реализуют архитектуру пиринговой сети. Такой подход обеспечит отказоустойчивость, передачу информации без посредников и возможность расширения системы без крупных вложений.

1. Модель данных системы

В данном разделе описывается модель данных автоматизированной системы обмена сообщениями на основе технологии пиринговой сети.

* 1. Стандарт функционального моделирования IDEF0

Чтобы подробнее изучить структуру и принципы функционирования системы была построена функциональная модель этой системы. Это позволяет сократить большое число ошибок на этапе раннего проектирования, тем самым улучшив качество проекта и уменьшив сроки его проектирования. Для построения модели использовалась система функционального моделирования IDEF0.

IDEF0 - это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между процессами, а не их временная последовательность.

Для более детального изучения структуры и функционирования информационной системы была построена ее модель. Построение модели исходной информационной системы позволяет сократить сроки проектирования и улучшить качество проекта за счет устранения большого числа ошибок уже на ранних этапах проектирования.

Для построения данной модели использовалась функциональная методология. Она предполагает рассмотрение системы в виде набора функций, преобразующих входной поток информации в выходной.

В России наибольшее распространение получила система функционального моделирования IDEF0 (IntegrationDEFinitionforFunctionModeling), являющаяся представителе семейства методологий IDEF.

IDEF0 - это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между процессами, а не их временная последовательность.

Описание информационной системы выглядит как «черный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня.

Модель информационного моделирования в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм.

Модель состоит из контекстной диаграммы и диаграмм декомпозиции. Контекстная диаграмма представляет общее описание системы и ее взаимодействие с внешней средой. Для одной системы контекстная диаграмма может быть только одна. Диаграммы декомпозиции описывают каждый фрагмент системы, полученной в ходе функциональной декомпозиции, и их взаимодействие. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так до достижения требуемого уровня детализации описания. Каждая диаграмма располагается на отдельном листе.

Графическим представлением процесса в методологии является функциональный блок, представляемый в форме прямоугольника с названием; поток, изображаемый в виде стрелки.

Поток отображает элемент системы, который:

1. обрабатывается функциональным блоком (входной поток);
2. является результатом обработки (выходной поток)
3. определяет правило получения правильного вывода (управляющий поток);
4. задает ресурсы, с помощью которых выполняется преобразование входа в выход (поток механизма).

Для разработки модели использовалось специализированное средство моделирования, поддерживающее нотацию IDEF0 – Microsoft Visio Professional.

* 1. IDEF0-модель автоматизированной системы обмена сообщениями на основе технологии пиринговой сети

Диаграмма функциональной модели информационной системы в методологии IDEF0 представлена на рисунках 2.1 - 2.5.

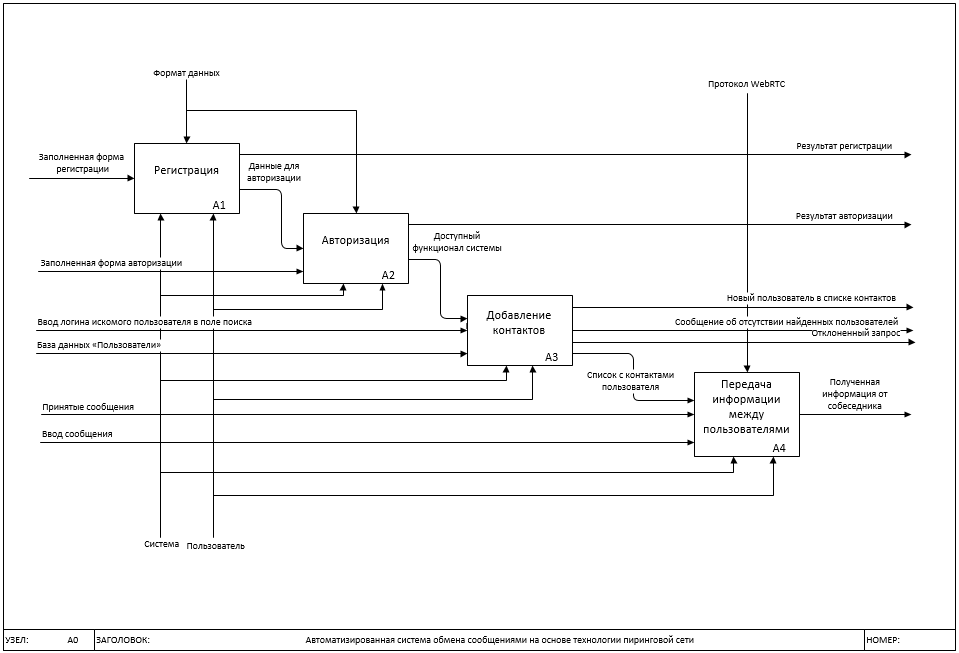


Рисунок 2.1 - Контекстная диаграмма работа автоматизированной системы обмена сообщениями на основе технологии пиринговой сети

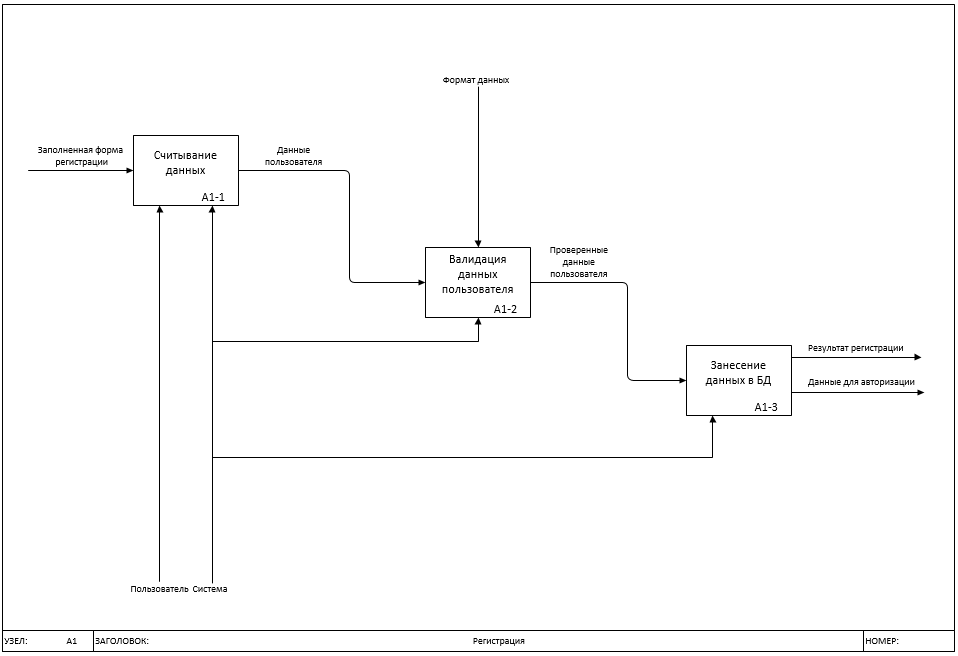


Рисунок 2.2 - Декомпозиция процесса «Регистрация»

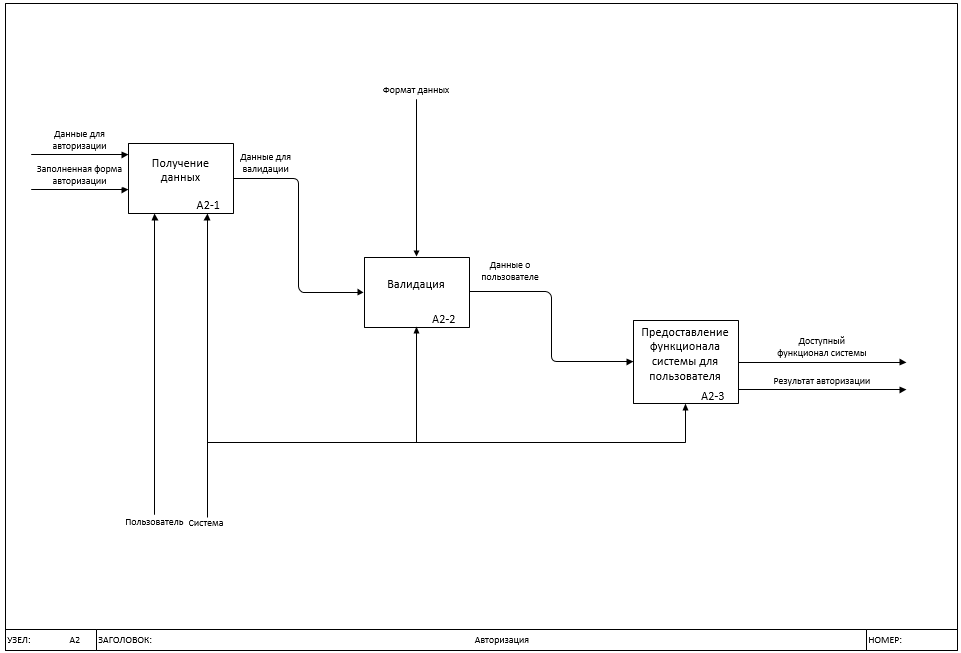


Рисунок 2.3 - Декомпозиция процесса «Авторизация»

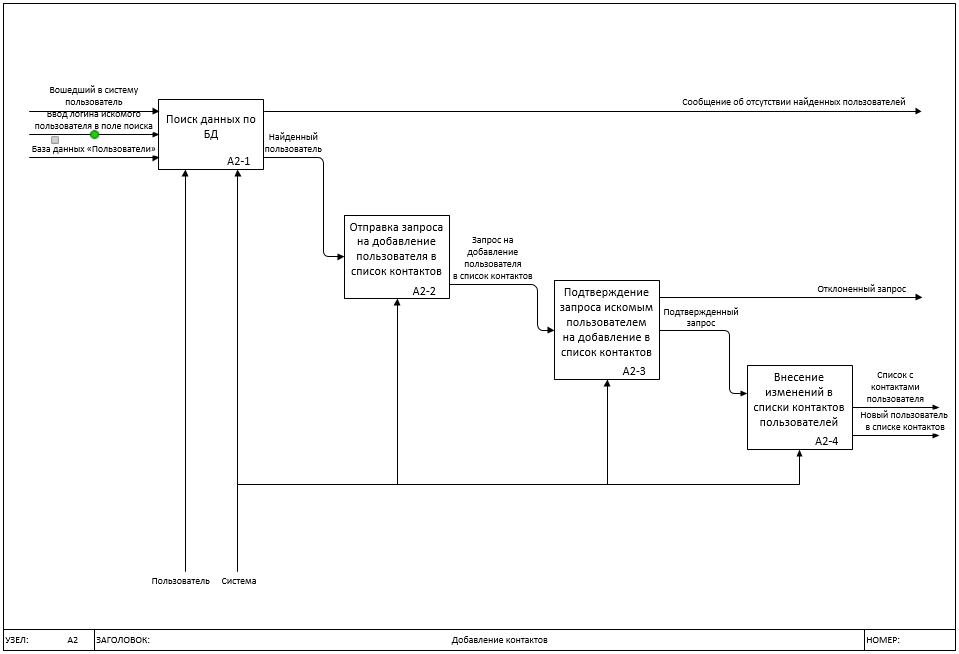


Рисунок 2.4 - Декомпозиция процесса «Добавление контактов»

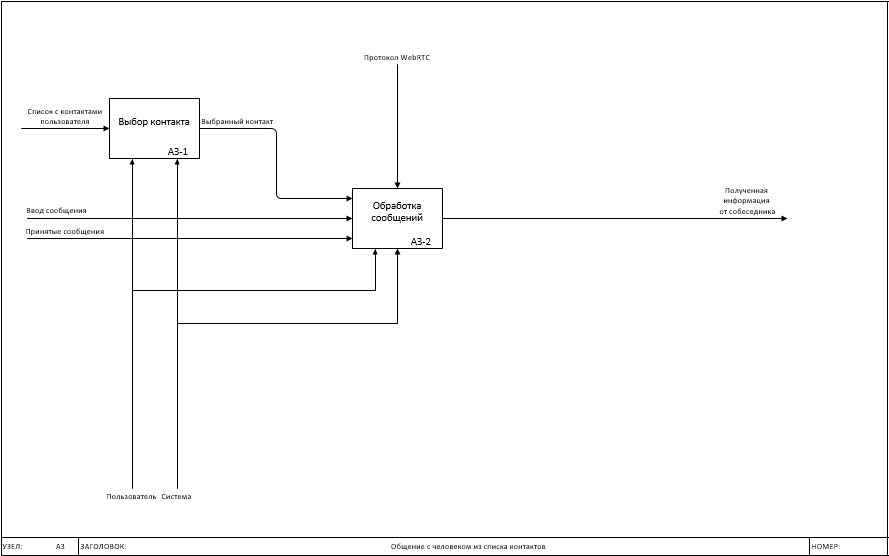


Рисунок 2.5 - Декомпозиция процесса «Общение с пользователем из списка контактов»

Декомпозиция процесса «Общение с пользователем из списка контактов» представленная на рисунке 2.5 демонстрирует, что процесс передачи информации в описываемой системе будет осуществляться посрдеством протокола WebRTC. Цель создания данной технологии — решение вопросов конфиденциальности при открытии локальных потоков. Безопасность и шифрование являются не дополнительными функциями WebRTC, а обеспечиваются встроенными компонентами по умолчанию. Кроме того, WebRTC предлагает сквозное шифрование между узлами практически на любом сервере, обеспечивая безопасную связь в режиме реального времени. Для передачи данных WebRTC использует протокол датаграмм безопасности транспортного уровня — [DTLS](http://www.3cx.ru/webrtc/dtls/)(Datagram Transport Layer Security). Этот протокол по умолчанию встроен во все браузеры, поддерживающие технологию WebRTC (Chrome, Firefox и Opera). В соединении, зашифрованном с помощью DTLS, исключается подслушивание и подделка информации.

Кроме DTLS, технология WebRTC использует для шифрования видео- и аудиоданных безопасный протокол передачи данных [SRTP](http://www.3cx.ru/webrtc/srtp/) (Secure Real-Time Protocol). Этот протокол исключает прослушивание или просмотр IP-связи (голосового и видео трафика) несанкционированными сторонами.

Любые действия в Интернете, например, загрузка VoIP-приложения (такого как Skype), загрузка фильма или даже передача информации или файлов по электронной почте, несут в себе риск злоумышленных вторжений. Однако технология WebRTC сохраняет конфиденциальность передаваемых конфиденциальных данных с помощью описанных выше протоколов и стандартов, обеспечивая безопасную связь в режиме реального времени.

С помощью технологии WebRTC компании имеют шанс трансформировать связь, предоставляя надежные и безопасные коммуникации корпоративного класса. По мнению аналитической компании Disruptive Analysis, к концу 2016 года количество индивидуальных пользователей WebRTC достигнет 1 миллиарда, а число смартфонов, планшетов и ПК с поддержкой WebRTC вырастет до 4 миллиардов. Несмотря на то, что первый успешный видеозвонок был совершенвсего в 2013 году, данная технология уже успела заслужить всеобщую популярность, а ее включение в рекомендации World Wide Web Consortium поддерживается [Google Chrome](https://ru.wikipedia.org/wiki/Google_Chrome), [Mozilla](https://ru.wikipedia.org/wiki/Mozilla_Foundation) и [Opera](https://ru.wikipedia.org/wiki/Opera_Software), что еще раз доказывает актуальность и перспективность технологий системы.

1. Информационное обеспечение системы

В данном разделе описывается информационное обеспечение подсистемы "Портал управляющих компаний" для АИС: Объектовый учёт.

* 1. Выбор технологий управления данными

В связи с высокой сложностьюИС для организации хранения и обработки данных была выбрана реляционная база данных.

Используемая при разработке информационной системы платформа .NET поддерживает множество транспортов данных от различных СУБД до прикладной программы.

Ввод данных в разрабатываемую систему должен осуществляться из других информационных систем. Некоторые поля могут быть вычислены автоматически при использовании соответствующих алгоритмов.

Согласно требованиям к информационному обеспечению (к реляционной СУБД) и лицензиям заказчика, был сделан выбор СУБД.Для удобства разработки и поддержки в дальнейшем был выбран Microsoft SQL Server [2]. Данная система управления базами данных уже работает на той же платформе .NET 4 или .NET 4.5, что и разрабатываемая информационная система. СУБД обладает полной документацией и работает под управлением операционной системы Microsoft Windows Server 2008 R2.

* 1. Проектирование базы данных
     1. Логическая и физическая модели данных

На рисунке 3.1 представлена ER-диаграмма логической модели данных в нотации IDEF1X.



Рисунок 3.1 – Логическая модель данных

Описание сущностей приведено в таблице 3.1.

Таблица 3.1 – Сущности логической модели данных

|  |  |
| --- | --- |
| Название сущности | Описание |
| 1. Объект | Сведения об объекте жилого фонда |
| 2. Строение | Сведения о строении жилого фонда |
| 3. Тип объекта | Сведения о типе объекта жилого фонда |
| 4. Элемент строения | Сведения об элементе строения |
| 5. Адрес | Подробные сведения об адресе |
| 6. Долгосрочная программа КР | Сведения о долгосрочной программе КР |
| 7. Документ | Основные сведения о документах |
| 8. Документ основания ДСП | Сведения о документе основанию на ДСП |
| 9. Протокол собрания собственников | Сведения о документе на проведенное собрание собственников жилья |
| 10. Организация | Подробная информация об организации |
| 11. Пользователь | Сведения о пользователе системы |
| 12. Должность | Сведения о должности для получения прав на редактирование информации о доме |

Список пользователей включён в логическую модель данных, так как информация о сотрудниках организаций необходима для ограничения доступа к изменению информации при первичном заполнении паспорта объекта и его редактировании. Ниже представлены таблицы с атрибутами сущностей. Дополнительно в таблицах приведено сопоставление логической и физической модели данных. Названия таблиц (в квадратных скобках) и полей сопоставлены с соответствующими сущностями и атрибутами.

Атрибуты сущности «Объект» представлены в таблице 3.2.

Таблица 3.2 – Атрибуты сущности «Объект», поля таблицы [Object]

| Название атрибута | Тип | Описание |
| --- | --- | --- |
| 1. Идентификатор объекта  {Id} | Числовой  {int} | Идентификатор объекта |
| 2. Описание  {Description} | Текстовый  {nvarchar(2000)} | Подробное описание объекта |
| 3. Количество этажей  {Floor} | Числовой  {int} | Количество этажей |
| 4. Дата постройки  {DateStart} | Дата  {datetime} | Дата постройки объекта |
| 5. Дата сноса  {DateEnd} | Дата  {datetime} | Дата сноса объекта |

Атрибуты сущности «Строение» представлены в таблице 3.3.

Таблица 3.3 – Атрибуты сущности «Строение», поля таблицы [Structure]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор строения  {Id} | Числовой  {int} | Идентификатор строения |
| 2. Описание  {Description} | Текстовый  {nvarchar(2000)} | Подробное описание строения |
| 3. Идентификатор типа объекта  {TypeId} | Числовой  {int} | Связь с типом объекта |
| 4. Идентификатор адреса  {AdressId} | Числовой  {int} | Связь с адресом |
| 5. Идентификатор объекта  {ObjectId} | Числовой  {int} | Связь с объеком |

Атрибуты сущности «Тип объекта» представлены в таблице 3.4.

Таблица 3.4 – Атрибуты сущности «Тип объекта», поля таблицы [ObjectType]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор типа объекта  {Id} | Числовой  {int} | Идентификатор типа объекта |
| 2. Код  {Code} | Текстовый  {nvarchar(50)} | Код типа объекта |
| 3. Название  {Name} | Текстовый  {nvarchar(200)} | Наименование типа объекта |

Атрибуты сущности «Элемент строения» представлены в таблице 3.5.

Таблица 3.5 – Атрибуты сущности «Элемент строения», поля таблицы [StructureComponent]

| Название атрибута | Тип | Описание |
| --- | --- | --- |
| 1. Идентификатор элемента строения  {Id} | Числовой  {int} | Идентификатор элемента строения |
| 2. Код  {Code} | Текстовый  {nvarchar(50)} | Код элемента строения |
| 3. Название  {Name} | Текстовый  {nvarchar(200)} | Наименование элемента строения |

Атрибуты сущности «Адрес» представлены в таблице 3.6.

Таблица 3.6 – Атрибуты сущности «Адрес», поля таблицы [Adress]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор адреса  {Id} | Числовой  {int} | Идентификатор адреса |
| 2. Населённый пункт  {Locality} | Текстовый  {nvarchar(50)} | Название населённого пункта |
| 3. Район  {Region} | Текстовый  {nvarchar(100)} | Название района |
| 4. Улица  {Street} | Текстовый  {nvarchar(50)} | Название улицы |
| 5. Дом  {Building} | Текстовый  {nvarchar(10)} | Номер дома |
| 6. Литера  {Literal} | Текстовый  {nvarchar(10)} | Литера дома |

Атрибуты сущности «Долгосрочная программа КР» представлены в таблице 3.7.

Таблица 3.7 – Атрибуты сущности «ДСП КР», поля таблицы [LongTermPlan]

| Название атрибута | Тип | Описание |
| --- | --- | --- |
| 1. Идентификатор долгосрочной программы  {Id} | Числовой  {int} | Идентификатор долгосрочной программы |
| 2. Год начала  {DateStart} | Числовой  {int} | Год начала действия долгосрочной программы КР |
| 3. Год окончания  {DateEnd} | Числовой  {int} | Год окончания действия долгосрочной программы КР |
| 4. Дата подтверждения  {ApprovedDate} | Дата  {datetime} | Дата подтверждениядолгосрочной программы КР |
| 5. Статус  {Status} | Текстовый  {nvarchar(50)} | Статус долгосрочной программы КР |

Также при переводе модели от логической до физической необходимо указать поля третьей сущности связи «Объект» - «Долгосрочная программа КР». Данные поля представлены в таблице 3.8.

Таблица 3.8 –поля таблицы [LongTermPlanObject]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор объекта  {ObjectId} | Числовой  {int} | Связь с объектом |
| 2. Идентификатор долгосрочной программы КР  {LongTermPlanId} | Числовой  {int} | Связь с долгосрочной программой КР |

Создание отдельных таблиц для описания сущностей «Документ основания ДСП» и «Протокол собрания собственников» было опущено ввиду увеличения скорости выборки данных информационной системой.

Атрибуты сущности «Документ» представлены в таблице 3.9.

Таблица 3.9 – Атрибуты сущности «Документ», поля таблицы [Document]

| Название атрибута | Тип | Описание |
| --- | --- | --- |
| 1. Идентификатор документа  {Id} | Числовой  {int} | Идентификатор документа |
| 2. Номер документа  {Number} | Текстовый  {nvarchar(50)} | Номер документа |
| 3. Имя файла  {Name} | Текстовый  {nvarchar(50)} | Имя файла |
| 4. Дата создания  {CreateDate} | Дата  {datetime} | Дата создания |
| 5. Описание  {Description} | Текстовый  {nvarchar(200)} | Описание |
| 6. Тип документа  {Type} | Текстовый  {nvarchar(50)} | Тип документа |
| 7. Организация управления  {OwnerOrganuzationId} | Числовой  {int} | Связь с организацией |

Атрибуты сущности «Организация» представлены в таблице 3.10.

Таблица 3.10 – Атрибуты сущности «Организация», поля таблицы [Organization]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор организации  {Id} | Числовой  {int} | Идентификатор организации |
| 2. Полное название организации  {FullOrganizationName} | Текстовый  {nvarchar(200)} | Полное название организации |
| 3. Краткое название организации  {ShortOrganizationName} | Текстовый  {nvarchar(50)} | Краткое название организации |
| 4. Телефон  {Phone} | Текстовый  {nvarchar(50)} | Телефон |
| 5. E-mail  {Contacts} | Текстовый  {nvarchar(200)} | E-mail |
| 6. ИНН  {INN} | Числовой  {int} | ИНН |
| 7. КПП  {KPP} | Числовой  {int} | КПП |
| 8. ОГРН  {OGRN} | Числовой  {int} | ОГРН |
| 9. Дата создания  {CreateDate} | Дата  {datetime} | Дата создания |
| 10. Идентификатор адреса  {AdressId} | Числовой  {int} | Связь с адресом |

Также при переводе модели от логической до физической необходимо указать поля третьей сущности связи «Объект» - «Организация», для однозначного определения принадлежности объекта управлению той или иной организации. Данные поля представлены в таблице 3.11.

Таблица 3.11 –поля таблицы [ObjectToOrganizationLink]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор объекта  {ObjectId} | Числовой  {int} | Связь с объектом |
| 2. Идентификатор организации  {OrganizationId} | Числовой  {int} | Связь с организацией |

Атрибуты сущности «Пользователь» представлены в таблице 3.12.

Таблица 3.12 – Атрибуты сущности «Пользователь», и часть полей таблиц[User]и [UserInformation]

| Название атрибута | Тип | Описание |
| --- | --- | --- |
| 1. Идентификатор пользователя  {Id} | Числовой  {int} | Идентификатор пользователя |
| 2. Идентификатор организации  {OrganizationId} | Числовой  {int} | Идентификатор организации |
| 3.Фамилия  {LastName} | Текстовый  {nvarchar(50)} | Фамилия |
| 4. Имя  {FirstName} | Текстовый  {nvarchar(50)} | Имя |
| 5. Отчество  {MiddleName} | Текстовый  {nvarchar(50)} | Отчество |
| 6. E-mail  {Contacts} | Текстовый  {nvarchar(200)} | E-mail |
| 7. Телефон  {Phone} | Текстовый  {nvarchar(50)} | Телефон |
| 8. Дата приёма  {CreateDate} | Дата  {datetime} | Дата приёма |
| 9. Дата увольнения  {LeaveDate} | Дата  {datetime} | Дата увольнения |
| 10. Логин  {Login} | Текстовый  {nvarchar(50)} | Логин для авторизации |
| 11. Пароль  {Password} | Текстовый  {nvarchar(50)} | Пароль для проверки авторизации |
| 12. Криптографический модификатор  {Salt} | Текстовый  {nvarchar(50)} | Так называемая «Соль», для усложнения защиты пароля |
| 13. Подтверждён  {IsApproved} | Логический  {bit} | Флаг подтверждения прав пользователя на работу с системой |
| 14. Идентификатор должности  {UserRoleId} | Числовой  {int} | Связь с должностью |

Атрибуты сущности «Должность» представлены в таблице 3.13.

Таблица 3.13 – Атрибуты сущности «Должность», поля таблицы [UserRole]

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип | Описание |
| 1. Идентификатор должности  {Id} | Числовой  {int} | Идентификатор должности |
| 2. Код  {Code} | Текстовый  {nvarchar(50)} | Код должности |
| 3. Наименование  {Name} | Текстовый  {nvarchar(200)} | Наименование должности |

* + 1. Проектирование реализации

Ввиду принятых на предприятии стандартов формирования БД для каждой таблицы были созданы представления. Из-за большого числа представлений и таблиц могла возникнуть путаница, для того чтобы этого не произошло название каждого представлениясовпадает с названием базовой таблицы и содержит префикс.Большинство представлений состоят из выборки данных из базовой таблицы, а также некоторых данных из связанных таблиц при помощи специальной конструкции«join» и «left join».В некоторых представлениях есть вложенные выборки на получение агрегированных данных. Также в представлении может существовать конструкция «case when else end», которая предназначена для получения данных на основании других полей.

* 1. Организация сбора, передачи, обработки и выдачи информации

В данном подразделе описываются основные действия связанные с информацией, такие как: сбор информации, передача информации, обеспечение достоверности информации, процедуры обслуживания и выдача информации.

* + 1. Сбор информации

В рамках реализации ИС нет требований к сбору информации, а соответственно и нет необходимости использовать технические средства сбора информации.

Вся информация, получаемая ИС от пользователя, может быть введена вручную и передана в ИС. Для осуществления данной передачи информации используется браузер (требования к нему указаны в разделе техническое задание).

* + 1. Передача информации

Передача информации в ИС производится при помощи всемирной системы объединённых компьютерных сетей Интернет. Для достижения этих целей используется протокол HTTPS, предполагающий клиент-серверную архитектуру общения программных продуктов: клиент отправляет запрос на сервер, сервер подготавливает ответ и отправляет его на клиент, клиент принимает ответ. Особенностью протокола является использование сертификатов для организации безопасного обмена данными между сервером и клиентом без возможности расшифровки перехваченной информации на каком-либо узле между адресатами (протокол HTTPS).

* + 1. Обеспечение достоверности

Достоверность введённой информации гарантируется протоколом HTTPS, использующим метод ассиметричного шифрования для предотвращения несанкционированного доступа к информации.

* + 1. Процедуры обслуживания

База данных, используемая информационной системой, архивируется ежедневно. В связи с тем, что при архивации базы данных информационная система перестаёт функционировать, рационально делать это в момент минимальной нагрузки на систему с 2 до 3 часов ночи. Данную задачу исполняет планировщик задач ОС Microsoft Windows.Во время публикации новой версии информационной системы, она также недоступна пользователям.

Закрытие и открытие доступа к системе происходит в автоматическом режиме и реализовано на основании настройки целей публикации приложения MSDeploy.

* + 1. Выдача информации

ИС не генерирует экранные формы. Она лишь предоставляетпользователю гипертекстовую разметку, в которой указывается гдеи как блоки на форме располагать и как их стилизовать. На полученной форме может быть как текстовая, так и графическая информация.

Выдача информации осуществляется на экран монитора. Информация в текстовом и бинарном виде приходит по каналам передачи информации к пользователю, а затем специальная программа (браузер) на основании полученных данных генерирует экранную форму.Эта программа нужна для корректной стилизации экранных форм.

1. Алгоритмическое обеспечение системы

В данном разделе рассматриваются алгоритмы, разработанные при создании подсистемы «Портал управляющих компаний» для АИС: Объектовый учет. Представленные алгоритмы разработаны с помощью ЯВУ C#.

* 1. Алгоритм автодополнения

Назначением алгоритма автодополнения данных является помощь пользователю при заполнении данных в поля для поиска дома или организации.

Структура алгоритма состоит из 2 частей.Первая часть представляет собой функцию, задача которой заключается в вызове подпрограммы для формирования списка автодополнения и передача ей необходимых входных параметров, после выполнения которой необходимо предоставить пользователю список возможных вариантов. Если были заполнены все необходимые поля на странице, то функции необходимо получить объект или организацию из БД и предоставить ссылку на него пользователю.

Вторая часть алгоритма представляет собой подпрограмму для получения объектов, удовлетворяющих условиям, на основании входных параметров, для формирования спискаавтодополнения. На вход подпрограмме подаётся исходная строка и префикс строки. Префикс строки необходим для определения уровня адреса, в случае если идёт формирование списка автодополнения для объекта. Исходная строка проходит проверку на заполненность, после чего происходит замена пробелов на знак “%”, также данный знак дописывается в начало и конец исходной строки. Это необходимо для формирования строки и поиска информации на её основании.

На рисунке 4.1 представлена схема подпрограммы первой части алгоритма, обеспечивающей вызов подпрограммы для формирования списка автодополнения и получения ссылки на объект или организацию.

Рисунок 4.1 – Схема алгоритма вызова подпрограммы для формирования списка автодополнения и получения ссылки на объект или организацию

На рисунке 4.2 представлена схема подпрограммы второй части алгоритма, для получения объектов и формирования списка автодополнения.



Рисунок 4.2 – Схема подпрограммы второй части алгоритма, для получения объектов и формирования списка автодополнения

Если при попытке формирования списка адресов окажется, что список элементов для строки автодополнения пуст, то пользователь получит информацию о том, что заданной фразе ничего не найдено. Данный момент не означает что пользователь сделал ошибку при введении информации в строку поиска, вполне возможно, что искомый объект просто отсутствует в системе.

* 1. Алгоритм фильтра по адресу

Назначением алгоритма фильтра по адресу является помощь пользователю при поиске дома в списке домов, участвующих в программе капитального ремонта. Необходимость фильтрации вызвана большим количеством объектов в системе, их более 20 тысяч.

Структура алгоритма фильтра по адресу представляет собой подпрограмму для получения объектов, удовлетворяющих условиям, на основании входных параметров, для упрощения поиска дома.

На вход подпрограмме подаётся исходная строка. Исходная строка проходит проверку на заполненность, после чего происходит замена пробелов на знак “%”, также данный знак дописывается в начало и конец исходной строки. Это необходимо для формирования строки для поиска и получение информации на её основании.

В случае если исходная строка пуста, то алгоритм получает список всех домов, участвующих в данной программе капитального ремонта. При отсутствии вхождений исходной строки, будет сформирован пустой список.

Отсортированный и сформированный список передаётся в модель для дальнейшего использования на странице со списком домов, участвующих в программе капитального ремонта.

На рисунке 4.3 представлена схема алгоритмафильтрации по адресу.



Рисунок 4.3 – Схема алгоритмафильтрации по адресу

1. Программное обеспечение системы

В разделе приводится описание инструментальных средств разработки, применяемых в процессе создания подсистемы, основных модулей ИС, руководства пользования подсистемой.

* 1. Выбор компонентов программного обеспечения

В подразделе приводится описание компонентов программного обеспечения, а также подробное описание инструментальных средств разработки.

В качестве технологии управления данными была выбрана СУРБДMicrosoftSQLServer(MSS). Причина такого выбора объясняется широким использованием технологии при проектировании и разработки баз данных для программных продуктов компании АИС Город.

Выбранная технология имеет как ряд преимуществ, так и недостатков. К положительным сторонам можно отнести:

1. СуществованиемногофункциональнойутилитыSQLServerManagementStudioдля администрирования, управления и конфигурирования всех компонентов MSS;
2. MSS – один из наиболее популярных продуктов компании Microsoft, поэтому в интернете помимо сайта официальной документации также существует множество форумов сообществ, посещение которых часто помогает разрешать нетривиальные вопросы, возникающие в процессе разработки;
3. Язык доступа к данным Transact-SQL,который является процедурным расширением языка SQL;
4. Бесплатная версия MSSобладает теми же возможностями и функционалом, что и профессиональная. Отличие заключается только в ограничении максимально возможного объема данных, равного 10-ти Гб, для свободной версии ПО.

Основными недостатками рассматриваемой технологии являются:

1. Для эффективного использования MSSнужно применять другие технологии программирования отфирмы Microsoft. Хотя в данном случае это скорее положительная сторона, так как проект объектового учета использует объектно-ориентированные технологии доступа к данным ADO.NETEntityFrameworkи LINQtoSQL;
2. Высокая стоимость полной версии MSS.
   * 1. Утилиты и инструменты проектирования базы данных

Для управления проектом базы данных использовался программный пакет SQLServerManagementStudio. Утилита представляет собой графическую среду, которая позволяет работать с объектами и настройками сервера базы данных. С ManagementStudio могут быть установлены расширения, значительно упрощающие процесс создания архитектуры базы данных и написания SQL-скриптов на выборку записей из таблиц. Примерами таких дополнений являются продукты компании RedgateSoftware:

1. SQLDataCompare, обеспечивающий автозаполнение при написании кодов SQL-запросов;
2. SQLMultiScript, позволяющий выполнять SQL-скрипт на нескольких выбранных базах данных одновременно.
   * 1. Технологии ORM

ObjectRelationalMapping (объектно-реляционное отображение) – это технология, позволяющая связать базу данных с концепцией объектно-ориентированного подхода в программировании. Основное назначение технологии ORM – это упрощение процессов управления (обновление, удаление или создание) объектами в реляционной базе данных. ORM самостоятельно занимается преобразованием данных между двумя несовместимыми состояниями: объектом в программном коде и записью в источнике данных.

Принцип работы большинства ORM-систем основан на метаданных базы данных и объектов. Таким образом, достигается полная независимость компонентов: объекты ничего «не знают» о структуре источника данных, а база данных – ничего о том, как эти объекты организованы и используются в программе.

Применение технологии ORM освобождает разработчика от необходимости работы с SQL-запросами и написания большого количества кода доступа к объектам в базе данных. Генерируемый ORM код для этих целей уже проверен и, как правило, хорошо оптимизирован, что, несомненно, является плюсом применения технологии. Однако основным недостатком использования ORM является потеря производительности. Это объясняется тем, что ORM-фреймворк поддерживает обработку широкого спектра сценариев управления данными, как правило, гораздо большего, чем предусмотрено в разрабатываемойИС.

Вопрос о применение ORM-технологий обычно затрагивается только при разработке крупных и высоконагруженных проектов, потому, что приходится выбирать между удобством и производительностью. Так как разработка данной ИС не является высоконагруженной, выбор пал на использование ORM технологии.

* + 1. Сравнение ORM-фреймворков LINQtoSQL и EntityFramework

Технологии от фирмы MicrosoftLINQtoSQL и EntityFramework предназначены для решения проблем объектно-реляционного отображения. Обе ORM-системы имеют свои особенности, сравнение которых приведено в таблице 5.1.

Таблица 5.1.- Сравнение возможностей ORM LINQ to SQL и Entity Framework

| Критерий | Entity Framework | LINQ to SQL |
| --- | --- | --- |
| Взаимодействие с СУБД | Может взаимодействовать с различными СУБД такими, как Oracle, DB2, MYSQL, Microsoft SQL Server и др. | Может взаимодействовать только с СУБД Microsoft SQL Server. |
| Типы данных | Поддерживает сложные типы данных. | Не поддерживает сложные типы данных. |
| Подход к созданию модели данных | Поддерживает методологию Model First. | Не использует методологию Model First. |
| Отношения | Позволяет отображать связи типов «один к одному», «один ко многим», «многие ко многим» между классами сущностей и реляционными таблицами/представлениями. | Позволяет только отображать связь типа «один к одному» между классами сущностей и реляционными таблицами/представлениями. |
| Обеспечение доступа к данным | Классы проводника доступа к данным – EntitySQL, ObjectContext, DbContext. | Базовый класс проводника доступа к данным – DataContext. |

Проведя сравнительный анализ ORM, выбор был сделан в пользу EF. Это решение объясняется более широким спектром возможностей при использовании данной ORM-технологии и тем фактом, что уже существующий к тому моменту проект модели данных функционировал на основе EF четвертой версии.

При разработке подсистемы возникла необходимость вызова в коде приложения хранимых процедур, написанных на языке Transact-SQL. Однако при попытке добавления процедуры в диаграмму ORMEF 4 по необъяснимым причинам не создавался тип объекта данных, возвращаемых процедурой.

Решение проблемы заключается в добавлении в реализацию хранимой процедуры «недосягаемой» части кода, в которой должны описываться метаданные возвращаемой временной таблицы.

При проведении процесса сопоставления источника данных и ORM-схемы, EF формирует на основе недоступной части кода процедуры объект сложного типа. В листинге 5.2 приведен пример разобранного случая.

Листинг 5.2 - Формирование сложного типа для EF на основе метаданных

--Для привязки данных в EntityFramework (недосягаемая часть кода)

IF 1 = 2 BEGIN

SELECT

CAST(NULL AS NVARCHAR(MAX)) AS ParameterName,

CAST(NULL AS NVARCHAR(MAX)) AS Value,

CAST(NULL AS INT) AS ArrayIndex

WHERE 1 = 2

END

--Создание временной таблицы

SET NOCOUNT ON;

CREATE TABLE #methodParameters ([Number] INT,

[ParameterName] NVARCHAR(MAX),

[ViewName] NVARCHAR(MAX),

[ViewColumnName] NVARCHAR(MAX),

[IsStructure] BIT,

[Value] NVARCHAR(MAX),

[ArrayIndex] INT,

[ExternalType] NVARCHAR(MAX))

--Результирующая выборка, используемая в коде приложения

SELECT ParameterName,

Value,

ArrayIndex

FROM #methodParameters

* 1. Разработка прикладного программного обеспечения

В данном подразделе приводится описание структуры разработанной подсистемы интеграции в целом и ее каждого модуля в отдельности.

* + 1. Структура прикладного программного обеспечения

Подсистема "Портал управляющих компаний" для АИС: Объектовый учёт состоит из двух частей: «Модуля в системе Объектовый учёт» и «Подсистемы портал управляющих компаний».

В рамках данной подсистемыможно выделить модули визуализации объекта, капитального ремонта, УК/ТСЖ/ЖСК, а также личного кабинета ПУК и модель объекта. Дробление системы на подсистемы и модули было выбрано ввиду явного разделения функций системы согласно техническому заданию.

Основной программный код подсистемы представлен в приложении А. Архитектура подсистемы «Портал управляющих компаний» представлена на рисунке 5.1.



Рисунок 5.1 – Архитектура подсистемы «Портал управляющих компаний»

Спецификация подсистемы «Портал управляющих компаний»указана в таблице 5.2.

Таблица 5.2 – Спецификация подсистемы «Портал управляющих компаний»

| Название компонента | Описание |
| --- | --- |
| Site | Личный кабинет ПУК |
| Main | Открытая часть подсистемы, доступная любому пользователю |

Таблица 5.3 – Спецификация модуляподсистемы «Объектовый учёт»

| Модули | |
| --- | --- |
| BigRepair | Модуль работы с капитальным ремонтом |
| HousingInspection | Модуль работы с ГЖИ |
| Personal | Модуль работы с объектами |
| ObjectDetails | Модуль работы с паспортом |
| Дополнительно подключаемые компоненты | |
| AIS.HM.Model | Модель данных ИС «Объектовый учёт». |

* + 1. Программный модуль «Портал управляющих компаний»

Спецификация модуля Siteуказана в таблице 5.4.

Таблица 5.4 – Спецификациямодуля Site

| Название компонента | Описание |
| --- | --- |
| public class ContentController | Содержит методы для предоставления информации об УК/ТСЖ/ЖСК |
| public class ContractController | Содержит методы предоставления информации о договорах |
| public classFinancialReportController | Содержит методы предоставления информации об оплате задолженностей |
| public classIndexController | Содержит методы предоставления общейинформации и сообщений |
| public classManagementContractController | Содержит методы предоставления информации об управлении контрактами |

Спецификация модуля Mainуказана в таблице 5.5.

Таблица 5.5 – Спецификациямодуля Main

| Название компонента | Описание |
| --- | --- |
| public classAccountController | Личный кабинет ПУК |
| public classAjxController | Открытая часть подсистемы, доступная любому пользователю |
| public class BigRepairController | Содержит методы для отображения страниц относящихся к капитальному ремонту |
| public class ErrorController | Содержит методы для отображения страниц с ошибкой |
| public class HelperController | Содержит методы для работы с Helper-ами |
| public class HomeController | Содержит методы для отображения главных страниц |
| public class HousesController | Содержит методы для отображения информации о доме |
| public classHousingInspectionController | Содержит методы для отображения деятельности государственной жилищной инспекции |
| public classMyAddressesController | Содержит методы для работы с «Моими адресами» |
| public classOrganizationController | Содержит методы для отображения информации об организациях |

* + 1. Программный модуль подсистемы «Объектовый учёт»

Спецификация модуля BigRepairуказана в таблице 5.6.

Таблица 5.6 – Спецификациямодуля BigRepair

| Название компонента | Описание |
| --- | --- |
| public classAccountController | Содержит методы работы с пользователями системы |
| public classDefectController | Содержит методы работы с дефектами строений |
| public classIndexController | Содержит методы работы с общей информацией |
| public class LongTermPlanController | Содержит методы работы с долгосрочными программами |
| public classRegionalOperatorController | Содержит методы для работы регионального оператора капитального ремонта |
| public class ShortTermPlanController | Содержит методы работы с краткосрочными программами |

Спецификация модуля HousingInspection указана в таблице 5.7.

Таблица 5.7 – Спецификациямодуля HousingInspection

| Название компонента | Описание |
| --- | --- |
| public classIndexController | Содержит методы работы с общей информацией |
| public classLicenseController | Содержит методы работы с лицензиями |
| public classRequestController | Содержит методы работы с ссылками возврата |
| public classVerificationController | Содержит методы для проверок договоров на капитальный ремонт. |

Спецификация модуля Personal указана в таблице 5.8.

Таблица 5.8 – Спецификациямодуля Personal

| Название компонента | Описание |
| --- | --- |
| public class AssociationController | Содержит методы для работы с СРО. |
| public classContractActualInfoController | Содержит методы для отчётности по работам капитального ремонта. |
| public classContractController | Содержит методы для работы с договорами на капитальный ремонт. |
| public classEmployeeController | Содержит методы работы с кадровым составом |
| public classDocumentController | Содержит методы работы с различными документами |
| public classOrgSiteInfoController | Содержит методы работы с информацией на сайте управляющих компаний |
| public classPageController | Содержит методы работы с пагинацией на страницах |
| public classStructureController | Содержит методы работы со строениями в системе |
| public classUserController | Содержит методы работы с учётной записью пользователей |

Спецификация модуля ObjectDetails указана в таблице 5.9.

Таблица 5.9 – Спецификациямодуля ObjectDetails

| Название компонента | Описание |
| --- | --- |
| public classApartmentDetailsController | Содержит методы работы с подробностями домов в управлении |
| public classElementController | Содержит методы работы с элементами объекта |
| public classEnergyCharacteristicController | Содержит методы работы с энергетическими характеристиками строения |
| public classPassportController | Содержит методы работы с паспортом объекта |
| public classStructureComponentsController | Содержит методы работы с элементами строения |
| public classStructureDetailsController | Содержит методы работы с подробностями элементами строения |

* 1. Инсталляция и особенности работы

Для работы с приложением необходима толькопрограмма (браузер) описанная в пункте 1.5.3 раздела технического задания, дополнительная инсталляцияпрограммного обеспечения на компьютеры пользователей не требуется.

* 1. Работа с основными разделами системы

Интерфейс пользователя – набор методов и средств взаимодействия информационной системы и пользователя.

Цель информационной системы упростить процесс работы с информацией об организациях, осуществляющих деятельность в сфере управления многоквартирными домами, а также предоставление различных сведений о многоквартирных домах и юридических лицах осуществляющих доверительное управление имуществом других физических и юридических лиц, переданным в соответствии с договором доверительного управления.

Можно выделить следующие страницы веб-интерфейсе информационной системы:

1. Страница авторизации;
2. Главная страница сменю;
3. Страница с перечнем «Моих адресов»;
4. Страница со списком управляющих компаний;
5. Страница со списком долгосрочных программ капитального ремонта и отчётами ревизионной комиссии;
6. Страница с перечнем данных по подготовке к зиме;
7. Страница с опубликованными планами проверок ГЖИ.

При открытии информационной системы в браузере пользователя встречает главная страница с меню (рисунок 5.2), предлагающая пользователю, для более комфортной работы с системой, пройти регистрацию и заполнить список интересующих его домов, чтобы не тратить время на их выбор при каждом посещении сайта.

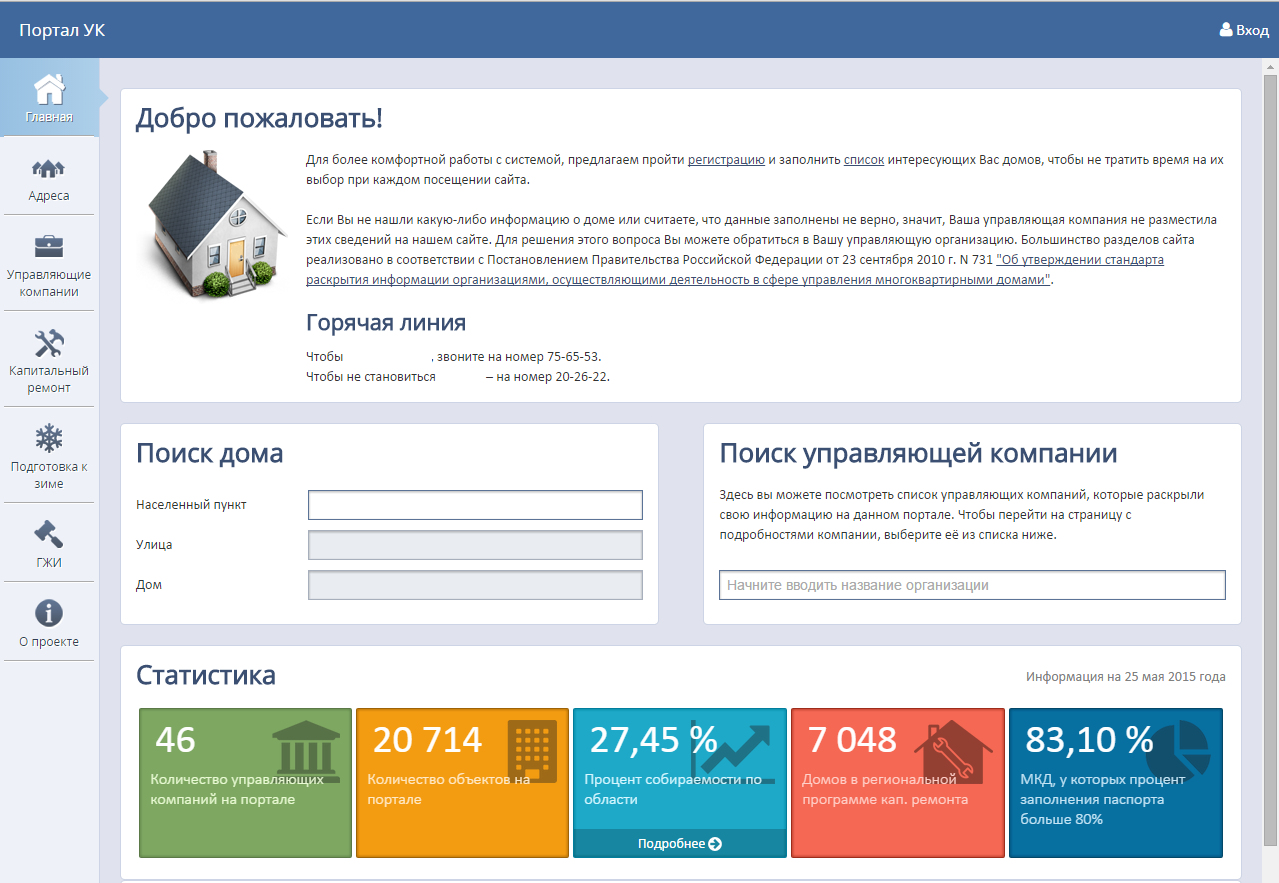


Рисунок 5.2 - Главная страница сменю.

При переходе на страницу с перечнем «Моих адресов» (рисунок 5.3), пользователю, не осуществившему вход в систему, предлагается добавить новый адрес, для отображения информации о доме. Для этого необходимо нажать кнопку «Добавить» на форме.

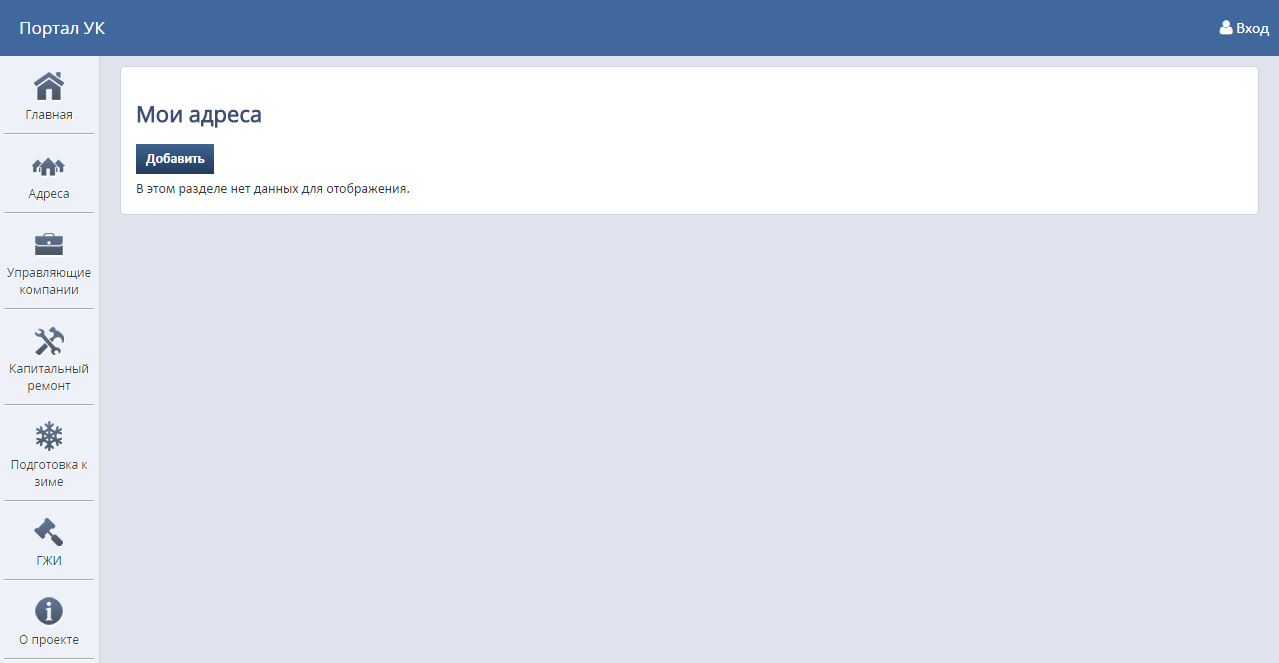


Рисунок 5.3 - Страница с перечнем «Моих адресов».

При переходе на страницу со списком управляющих компаний (рисунок 5.4), пользователю предоставляется краткая информация об УК/ТСЖ/ЖСК: название, юридический адрес, телефон. Для ознакомления с полной и подробной информацией об организации, необходимо перейти по ссылке располагающейся в названии организации.



Рисунок 5.4 - Страница со списком управляющих компаний.

При переходе на страницу со списком долгосрочных программ капитального ремонта и отчётами ревизионной комиссии (рисунок 5.5), пользователю предоставляется краткая информация о периоде и статусе долгосрочной программы, а также прикреплённые документы ревизионной комиссии.

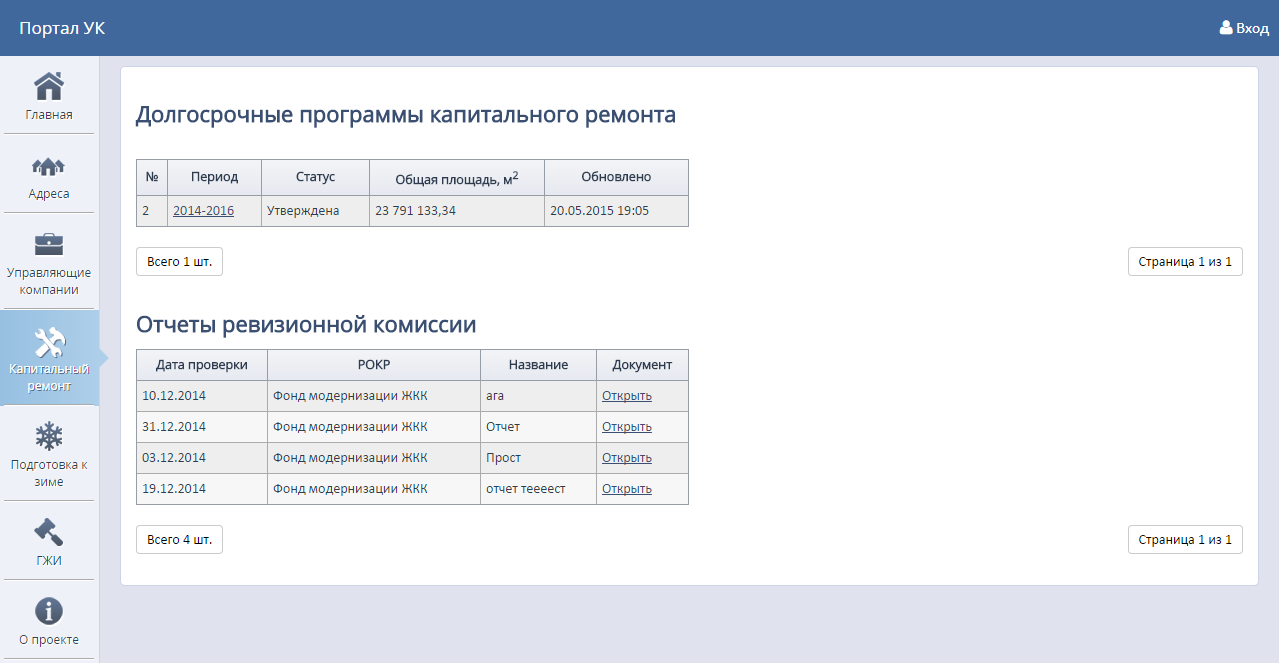


Рисунок 5.5 - Страница со списком долгосрочных программ капитального ремонта и отчётами ревизионной комиссии.

При переходе на страницу с опубликованными планами проверок ГЖИ (рисунок 5.6), пользователю предоставляется краткая информация о плане: год, статус, даты согласования, поверки и публикации. Для ознакомления с подробностями плана проверки, необходимо перейти по ссылке «Подробности» располагающейся в первой колонке.

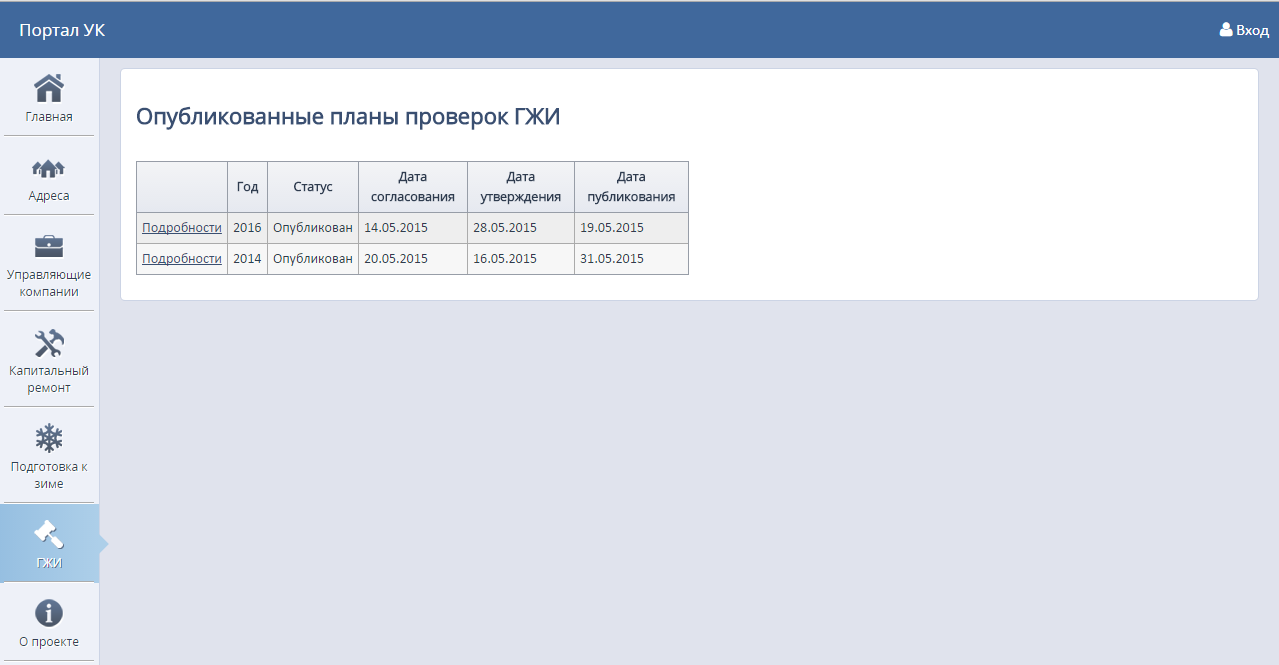


Рисунок 5.6 - Страница с опубликованными планами проверок ГЖИ.

1. Тестирование системы

В данном разделе описываются основные моменты тестирования автоматизированной системы обмена сообщениями на основе технологии пиринговой сети.

* 1. Условия и порядок тестирования

Объектом тестирования является ИС. В качестве методов тестирования были выбраны две модели: метод «белого ящика» и метод «черного ящика». В основе методологии «черного ящика» лежит принцип того, что имеют значения только входные и выходные данные. Используется ручной тип проверки функционирования системы. Метод же «белого ящика» необходим для выявления проблем с взаимодействием компонентов системы.

Общий порядок тестирования для тестирования методом «чёрного ящика»:

1. Снимаются входные данные;
2. Снимаются выходные данные.

Общий порядок тестирования для тестирования методом «белого ящика»:

1. Снимаются входные данные;
2. Пошагово проверяется работа всех компонентов ИС;
3. Снимаются выходные данные.

Тип проверки функционирования системы – ручной.

Основной упор при тестировании делается на проверку правильной отрисовки интерфейса пользователя и правильной работы системы в случае указания некорректных входных данных.

* 1. Исходные данные для контрольных примеров
     1. Форма авторизации

Пользователь вводит логин и пароль в специальные формы, нажимает кнопку «Войти».

Варианты поведение пользователя в системе:

* данные о пользователе введены корректно и произведен вход в ИС;
* пользователь указал неверные данные для авторизации (рисунок 6.1); Другое поведение пользователя в системе рассматривается как ошибочное и считается что тест не пройден.
  + 1. Страница приложения

На странице приложения пользователю предлагается воспользоваться поиском для добавления новых контактов. Для поиска контактов предусмотрено поле поиска (результат представлен на рисунке 6.2).

На данном этапе задачей тестирования являлось проверка корректности работы алгоритма добавления контакта в список контактов пользователя. Если существующий в базе данных пользователь не добавляется в список контактов, то тест считается непроденным.

* 1. Результаты тестирования

Объектом автоматизации информационной системы является процесс передачи информации напрямую от одного пользователя другому.

Тестирование подсистемы показало, что после доработок по окончании тестирования, основной функционал системы реализован и работает исправно. На данный момент проверялось правильное отображение интерфейса веб-приложения системы, сама система работоспособна и соответствует требованиям, предъявляемым ей техническим заданием на разработку.

1. Экономический раздел

Основным содержанием данного раздела является технико-экономическое обоснование проекта, т. е. определение экономической эффективности процессов создания и внедрения проектируемой системы в эксплуатацию.

* 1. Расчет показателя трудоемкости для программного продукта

Трудоемкость работ — это показатель, характеризующий затраты живого труда, выраженные в рабочем времени, затраченном на производство продукции или услуг. Величина данного параметра напрямую зависит от продолжительности периодов времени, занимаемых каждым из этапов проектирования программного продукта.Чтобы выполнить разработку интеграционного слоя информационных систем необходимо начать с анализа предметной области, в которой будет использоваться создаваемый программный продукт.

После детального изучения сферы применения наступает время процесса прогнозирования временных затрат для каждого из этапов проектирования. Подходить к этим расчетам нужно ответственно, чтобы свести к минимуму погрешности в оценке трудоемкости работ по проекту.

В настоящее время для определения трудоемкости разработки информационных приложений применяется способ оценки работ в человеко-часах. Эффективность методики подтверждена ведущими современными IT-компаниями.

Величина параметра трудоемкости для разрабатываемого программного решения состоит из суммы значений трудоемкости для каждого этапа разработки и рассчитывается по формуле 7.1n:

, (7.1)

где - общая трудоемкость разработки программного продукта,

- трудоемкость разработки i-го этапа проектирования,

n - общее количество этапов проектирования.

Проанализировав формулу, напрашивается вывод о том, что если проект разделен на большее количество стадий разработки, то искомая оценка трудоемкости выполняемых работ будет точнее. В таблице 7.1 приведены данные о расчете величины параметра трудоемкости для каждого из этапов проектирования и для всего проекта в целом.

Таблица 7.1 **-** Поэтапная и общая оценка трудоемкости программного решения

| Этап разработки | Вид работ | Длительность работ (чел. \* час.) |
| --- | --- | --- |
| Формирование требований к системе | Исследование предметной области объекта проектирования. Анализ требований пользователей | 20 |
| Разработка технического задания | Написание документов технического задания на систему. Утверждение технического задания | 60 |
| Изучение принципов и методик работы с информацией | Выбор методики обработки данных и их хранение. | 60 |
| Реализация программного решения | Разработка информационного продукта на языке программирования | 300 |
| Развёртка решения на сервере | Развёртка разработанного программного решения на тестовом сервере | 30 |
| Тестирование системы | Проведение тестирования разработанного программного решения на тестовых данных. Устранение ошибок  Проведение мер по тестированию с использованием реальных данных. | 40 |
| Написание рабочей документации | Разработка сопроводительной документации на систему | 50 |
| Процесс внедрения | Поставка готового решения пользователям | 10 |
| Итого: |  | 570 |

В таблице 7.2 представлен график проведения работ по проекту.

Таблица 7.2 - График проведения работ по проекту

| Вид работ | Исполнитель | Трудоемк., чел.-час. | Кол-во дней | Продолжительность работы | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|
| 4 | 14 | 12 | 40 | 3 | 5 | 7 | 2 |
| Формирование требований к системе | Инженер - программист | 20 | 4 |  |  |  |  |  |  |  |  |
| Разработка технического задания | Инженер - программист | 60 | 14 |  |  |  |  |  |  |  |  |
| Изучение принципов и методологий работы с информацией | Инженер - программист | 60 | 12 |  |  |  |  |  |  |  |  |
| Реализация программного решения | Инженер - программист | 300 | 40 |  |  |  |  |  |  |  |  |
| Развёртка решения на сервере | Инженер - программист | 30 | 3 |  |  |  |  |  |  |  |  |
| Тестирование системы | Инженер – программист, сотрудник отдела QA, заказчики системы | 40 | 5 |  |  |  |  |  |  |  |  |
| Написание рабочей документации | Инженер - программист | 50 | 7 |  |  |  |  |  |  |  |  |
| Процесс внедрения | Инженер-программист | 10 | 2 |  |  |  |  |  |  |  |  |
| Общая трудоемкость и длительность проведения работ по проекту | | 570 | 87 |  | | | | | | | |

* 1. Расчет затрат на материальные ресурсы и сырье

Материальные ресурсы – это различные виды сырья, материалов, топлива, энергии, комплектующих и полуфабрикатов, которые хозяйствующий субъект закупает для использования в хозяйственной деятельности с целью выпуска продукции, оказания услуг и выполнения работ.

Процесс проектирования выпускной квалификационной работы бакалавра требовал вычислить определенный ресурс в виде материальных и сырьевых затрат. Расчет стоимости необходимых материалов производился с помощью формулы 7.2:

, (7.2)

где - расход i-го вида материального ресурса, натуральные единицы,

- цена за единицу i-го вида материального ресурса,

i - вид материального ресурса,

n - общее количество всех видов материальных ресурсов.

Результаты расчетов затрат на материальные ресурсы приведены в таблице 7.3.

Таблица 7.3 **-** Сумма затрат на материальные ресурсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Единица измерения | Количество израсходованного материала | Цена за единицу, руб. | Сумма, руб. |
| Ноутбук | шт. | 1 | 22700,00 | 22700,00 |
| Принтер | шт | 1 | 9800,00 | 9800,00 |
| Канцелярские принадлежности | шт | 1 | 300,00 | 300,00 |
| Полная сумма затрат на материальные ресурсы | | | | 32 800,00 |

Общая стоимость расходных материалов рассчитывается также по формуле. Необходимые расчеты стоимости затрат на расходные материалы отображены в 7.4.

Таблица 7.4 **-** Расчет стоимости затрат на расходные материалы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Единица измерения | Количество израсходованного материала | Цена за единицу, руб. | Сумма, руб. |
| Оплата услуг интернет-провайдера | руб./мес. | 4 | 400 | 1600 |
| Бумага для принтера | упак. | 1 | 300 | 300 |
| Полная сумма затрат на материальные ресурсы | | | | 1900 |

Расчет стоимости затраченной электроэнергии в процессе написания дипломного проекта производится на основе действующих тарифов на электроэнергию, устанавливаемых региональными энергетическими комиссиями. Общая сумма энергетических затрат рассчитывается по формуле 7.3:

, (7.3)

где- сумма затрат на электроэнергию,

- паспортная мощность электрооборудования i-го вида, измеряется в кВт,

Ц - тариф электроэнергии, руб./кВт \* ч.

i - вид прибора электрооборудования,

n - общее число электроприборов.

Необходимые расчеты затрат на электроэнергию приведены в таблице 7.5.

Таблица 7.5 **-** Расчет затрат на электроэнергию

| Наименование единицы оборудования | Паспортная мощность, кВт. | Время работы оборудования, ч | Тариф электроэнергии, руб/кВт \* ч | Сумма, руб |
| --- | --- | --- | --- | --- |
| Ноутбук | 0,25 | 570 | 2,9 | 413,25 |
| Принтер | 0,1 | 1 | 0,29 |
| Освещение рабочего места | 0,04 | 210 | 24,36 |
| Итого за электроэнергию | | | | 437,9 |

* 1. Расчет затрат на разработку системы

Определение затрат на разработку производится путем составления соответствующей сметы, которая включает следующие статьи:

* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных фондов.
  1. Расчет затрат на оплату труда

Общая сумма затрат на оплату труда определяется по формуле 7.4:

, (7.4)

где Зп– общая сумма затрат на оплату труда, руб.;

– часовая ставка i-го работника, руб.;

– время на разработку системы, час;

i – категория работника;

n – количество работников, занятых разработкой системы.

Расчеты заработных отчислений приведены в таблице 7.6.

Таблица 7.6 **-** Расчет заработной платы сотрудников

| Квалификация сотрудника | Трудоемкость, чел.-час. | Часовая ставка, руб. | Сумма, руб. |
| --- | --- | --- | --- |
| Инженер-программист | 570 | 120,00 | 68 400,00 |
| Итого затрат на начисление заработных плат | | | 68 400,00 |

* 1. Расчет отчислений на социальные нужды

Данные об отчислениях на социальные нужды представлены в таблице 7.7.

Таблица 7.7 - Отчисления на социальные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Вид | Начислено заработной платы, руб. | Отчисления, % | Сумма, руб. |
| Фонд социального страхования РФ | 68 400,00 | 30 | 20 656,80 |
| Профессиональные заболевания | 0,2 | 136,80 |
| Итого на социальные нужды | | | 20 793,60 |

* 1. Расчет амортизационных отчислений

В ходе изготовления программного продукта использовалось оборудование (ноутбук) общей стоимостью 22700,00 руб.. Общая сумма амортизационных отчислений определяется по формуле 7.6:

, (7.6)

где ЗАМ – общая сумма амортизационных отчислений, руб.;

Фi – стоимость i-го оборудования, руб.;

HAi – годовая норма амортизации i-го оборудования, %;

TНИРi – время работы i-го оборудования за весь период разработки, ч;

TЭфi – эффективный фонд времени работы i-го оборудования за год, ч/год;

i – вид оборудования;

n – количество оборудования.

Общая сумма амортизационных отчислений равна:

(22 700 10 570) / (100 1800) = 718,83 руб.

* 1. Себестоимость проекта

Данные о себестоимости проекта представлены в таблице 7.8.

Таблица 7.8 - Себестоимость проекта

|  |  |
| --- | --- |
| Статьи затрат | Сумма, руб. |
| Затраты на материальные ресурсы | 32 800,00 |
| Затраты на оплату труда | 68 400,00 |
| Отчисления на социальные нужды | 20 793,60 |
| Амортизация основных фондов | 718,83 |
| **Итого по смете:** | 122 712,43 |

* 1. Расчет показателей экономической эффективности и ожидаемого годового экономического эффекта от внедрения разработки

Экономический эффект определяется как разность между годовой экономией (или годовым приростом) и нормативной прибылью. Годовой экономический эффект Э, руб, рассчитывается по формуле 7.7:

, (7.7)

где П – годовая экономия (или годовой прирост), руб.(рассчитывается по формулам 7.8 – 7.9);

К – единовременные затраты, руб. (рассчитывается по формуле 7.10);

Ен – нормативный коэффициент эффективности капитальных вложений (Ен = 0,15).

, (7.8)

, (7.9)

где Зразр – затраты на разработку системы, руб.;

Р – средний уровень рентабельности, % (принимается в размере 20-30%).

, (7.10)

где Зразр – затраты на разработку системы, руб.;

Кк – капиталовложения в комплект технических средств.

Ц = 122 712,43 (1 + 20 / 100) = 147 254,92

П = 147 254,92 - 122 712,43 = 24 542,49

Э = 24 542,49 - 0,15 122 712,43 = 6 135,62

Расчетный коэффициент эффективности капитальных вложений(рассчитывается по формуле 7.11):

Ер = П / К , (7.11)

24 542,49 / 122 712,43 = 0,19

Ер> Ен, что значит, что капитальные затраты можно считать целесообразными.

Срок окупаемости проекта (рассчитывается по формуле 7.12):

Т = К / П , (7.12)

122 712,43 / 24 542,49 = 5 месяцев.

* 1. Расчет плановой прибыли

От того, насколько достоверна определена плановая прибыль, будет зависеть успешная финансово-хозяйственная деятельность предприятия. Расчет плановой прибыли должен быть экономически обоснованным. Это позволит осуществлять своевременное и полное финансирование инвестиций, прироста собственных оборотных средств и соответствующих выплат сотрудникам.

Плановая прибыль реализации программного решения рассчитывается по формуле 7.13:

, (7.13)

где- полная себестоимость, руб.,

- норматив рентабельности.

При нормативе рентабельности, равном 30%, прибыль будет составлять 36061,071 руб. С учетом налога на прибыль, составляющим 20 %, доход составит:

1. Безопасность и экологичность проекта

В данном разделе приводится описание безопасности и экологичности проекта – автоматизированной системы обмена сообщениями на основе пиринговой сети.

* 1. Исходные данные

В таблице 8.1 приведены исходные данные для проектирования.

Таблица 8.1 - Исходные данные для проектирования

| Наименование | Фактическое значение |
| --- | --- |
| Тема дипломного проекта | Разработка автоматизированной системы обмена сообщениями на основе пиринговой сети. |
| Вид технологического процесса | Проектирование программного продукта |
| Вид оборудования, паспортные данные | Компьютер, принтер |
| Напряжение, режим нейтрали электрической сети | 220 В, 50 Гц, с заземлением |
| Характеристика производственного помещения по электроопасности | Согласно ГОСТ 12.1.019-79, электрооборудование помещения относится к 1 классу защиты от поражения электрическим током, т.е. имеется рабочая изоляция, элемент для заземления и провод без зануляющей шины для подсоединения к источнику питания. |
| Характеристика среды помещения | Допустимые показатели микроклимата помещения соответствуют ГОСТ 12.1.005-88. Уровень звукового давления (45 дБ) меньше максимально допустимого уровня (согласно СанПиН 2.2.2./2.4.1340-03, допустимый уровень звукового давления при работе на ВДТ (видеодисплейный терминал) и ПЭВМ не должен превышать 60 дБ). |

Окончание таблицы 8.1

|  |  |
| --- | --- |
| Наименование | Фактическое значение |
| Признаки отнесения объекта проектирования к опасным  объектам | Нет |
| Категория производства по взрывопожарной опасности | Помещение можно отнести к категории «В» (ОНТП 24-86) (помещение содержит горючие и трудногорючие жидкости, твердые горючие и трудногорючие вещества в малом количестве и материалы способные только гореть при взаимодействии с кислородом воздуха). |
| Характеристика взрывопожароопасных зон | По классификации класс пожароопасных зон помещение относится к П-2-А (зона, в которой обращаются твердые горючие вещества) согласно ПУЭ (правила устройства электроустановок). |
| Категория взрывоопасных смесей | Нет |
| Профессия рабочего, эксплуатирующего объект проектирования | Технические специалисты, работники УК/ТСЖ/ЖСК обученные работе с системой. |
| Классы условий труда в соответствии с картой аттестации рабочего места:  по вредности,  по травмоопасности | класс 3.1 – вредный  класс 2 – допустимый |

* 1. Перечень нормативных документов

Перечень нормативных документов:

* Санитарные правила и нормы. СанПиН 2.2.2./2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.
* «Руководство по гигиенической оценке факторов рабочей среды и трудовых процессов. Критерии и классификация условий труда». Р 2.2.2006-05.
* ГОСТ 12.0.003-74.ССБТ. (СТ СЭВ 790-77) Опасные и вредные производственные факторы. Классификация. М.: Изд-во стандартов, 1996.
* ГОСТ 12.1.003-83.ССБТ. Шум. Общие требования безопасности. М.: Изд-во стандартов.1996.
* ГОСТ 12.1.004-91.ССБТ. Пожарная безопасность. Общие требования. М.: Изд-во стандартов, 1996.
* ГОСТ 12.1.005-88.ССБТ. Общие санитарно-гигиенические требования к воздуху рабочей зоны. М.: Изд-во стандартов, 1996.
* ГОСТ 12.1.006-88.ССБТ. Электромагнитные поля радиочастот. Допустимые уровни на рабочих местах и требования к проведению контроля. М.: Изд-во стандартов, 1998.
* ГОСТ 12.1.019-79.ССБТ (СТ СЭВ 4880-84). Электробезопасность. Общие требования. М.: Изд-во стандартов, 1996.
* ГОСТ 12.1.030-81.ССБТ. Электробезопасность. Защитное заземление зануление. М.: Изд-во стандартов, 1996.
* ГОСТ 12.1.038-82.ССБТ. Электробезопасность. Предельно-допустимые значения напряжений прикосновения и токов. М.: Изд-во стандартов, 1996.
* Общесоюзные нормы технологического проектирования ОНТП 24-86., М.: МВД СССР, 1986.
* СНиП 2.01.02-85. Противопожарные нормы. М.: Стройиздат,1986.
* СНиП 2.04.05-86. Отопление, вентиляция, кондиционирование воздуха. М.: Стройиздат, 1988.
* СНиП 23-05-95. Естественное и искусственное освещение. Анализ проектирования. М.: Энерго, 1996.
* Р 2.2.013-94. Гигиена труда. М.: Госкомсанэпиднадзор России, 1994.
* Правила пожарной безопасности в Российской Федерации – ППБ 01 03.
* Нормы пожарной безопасности – НПБ 88-2001. Установки пожаротушения и сигнализации. Нормы и правила проектирования.
  1. Анализ потенциальных опасностей

На рисунке 8.1, приведена принципиальная блок-схема обеспечения безопасности объекта проектирования.



Рисунок 8.1 - Принципиальная схема обеспечения безопасности объекта

проектирования

* + 1. Анализ вредных и опасных производственных факторов

Опасный производственный фактор – это производственный фактор, воздействие которого в определенных условиях приводит к травме или к другому внезапному ухудшению здоровья.

Воздействие вредного производственного фактора в определенных условиях приводит к заболеванию или снижению работоспособности.

Классификация опасных и вредных производственных факторов (ГОСТ 12.0.003-74). Опасные и вредные производственные факторы подразделяются по природе действия на следующие группы: химические, биологические, психофизические, физические.

Все факторы, за исключением психофизических обусловлены воздействием техники и рабочей среды. Психофизиологические факторы связаны с влиянием тяжести и напряженности труда, что в конечном итоге тоже может привести к заболеваниям.

Так как на рабочем месте, рассматриваемом в рамках данного дипломного проекта, химические и биологические опасные и вредные производственные факторы оказывают незначительное, по сравнению с физическими факторами, влияние, в рассмотрение они браться не будут.

При работе с ПЭВМ на пользователя в той или иной степени могут воздействовать следующие физические факторы: повышенные уровни переменного электромагнитного и электростатического полей; повышенный уровень статического электричества; повышенный уровень низкоэнергетического (мягкого) рентгеновского ионизирующего излучения; повышенные уровни ультрафиолетового и инфракрасного излучения; повышенное содержание положительных аэроионов в воздухе рабочей зоны; пониженное содержание отрицательных аэроионов; аномальный уровень освещённости рабочей зоны; повышенная яркость фрагментов светового изображения или света, попадающего в поле зрения пользователя; повышенная неравномерность распределения яркости в поле зрения пользователя; повышенная внешняя освещённость экрана; повышенные пульсации светового потока источников света или светового потока, излучаемого экраном; неблагоприятный для работы спектр излучения источников света; повышенная временная нестабильность изображения; мерцание экрана; изменение яркости свечения экрана; повышенная прямая блескость, вызванная попаданием в поле зрения работающего чрезмерно яркого света различных излучающих объектов; повышенная отражённая блескость, обусловленная наличием зеркальных отражений (бликов), в том числе от экрана; повышенный уровень шума; аномальные температура, влажность и подвижность воздуха рабочей зоны; повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека; пожар.

* + - 1. Шум

Шум является общебиологическим раздражителем и в определенных условиях может влиять на все органы и системы организма человека. Кроме непосредственного воздействия на орган слуха шум влияет на различные отделы головного мозга, изменяя нормальные процессы высшей нервной деятельности. Это так называемое неспецифическое воздействие шума может возникнуть даже раньше, чем изменения в органе слуха. Шумовые явления обладают свойством аккумуляции: накапливаясь в организме, он все больше и больше угнетает нервную систему. Шум – причина преждевременного утомления, ослабления внимания, памяти.

Согласно СанПиН 2.2.2./2.4.1340-03, допустимым уровнем звукового давления при работе на ВДТ и ПЭВМ не должно превышать 60 дБ. Экспериментальные данные показывают, что уровень звукового давления (33 дБ) меньше предельно допустимого уровня.

Мероприятия по защите от шума, проводимые в производственном помещении соответствуют ГОСТ 12.1.003-83 и других мероприятий по улучшению шумовой обстановки не требуется.

* + - 1. Микроклимат

Микроклимат помещений - это климат внутренней среды помещений, который определяется действующими на организм человека сочетаниями температуры, влажности и скорости движения воздуха, а также температуры окружающих поверхностей.

Показателями, характеризующими микроклимат в производственном помещении, являются:

1. температура воздуха;
2. относительная влажность воздуха;
3. скорость движения воздуха;
4. интенсивность теплового излучения.

Значительное отклонение микроклимата рабочей зоны от оптимального может быть причиной ряда физиологических нарушений в организме работающих, привести к резкому снижению работоспособности и даже к профессиональным заболеваниям.

В помещениях с вычислительной техникой при выполнении работ операторского типа, связанных с нервно-эмоциональным напряжением, по ГОСТ 12.1.005-88 необходимо соблюдать оптимальные величины показателей:

1. температура помещения в переходный период 18 – 22°С, в холодный период 22 – 24°С, в теплый период 20 – 24°С;
2. подвижность воздуха – от 0,1 до 0,2 м/с;
3. влажность воздуха составляет 60 – 70%;
4. воздействие химических веществ отсутствует;
5. запыленности и загазованности воздуха нет;
6. выполняются легкие физические работы (1 категория).

Колебания температуры воздуха допускаются до 4%.

Для создания нормальных условий труда в производственных помещениях обеспечивают нормативные значения параметров микроклимата – температуры воздуха, относительную влажность и скорость движения, а также интенсивности теплового излучения.

В ГОСТ 12.1.005-88 указаны оптимальные и допустимые показатели микроклимата в производственных помещениях. Оптимальные показатели распространяются на всю рабочую зону, а допустимые устанавливают раздельно для постоянных и непостоянных рабочих мест в тех случаях, когда по технологическим техническим или экономическим причинам невозможно обеспечить оптимальные нормы.

Мероприятия по обеспечению оптимальных метеоусловий соответствуют ГОСТ 12.1.005-88 и СНиП 2.04.05-86 и других мероприятий по обеспечению микроклимата не требуется.

* + - 1. Электрический ток

Опасное и вредное воздействие на людей электрического тока проявляется в виде электротравм и профессиональных заболеваний. Степень опасного и вредного воздействий на человека электрического тока зависит от:

1. рода и величины напряжения и тока;
2. частоты электрического тока;
3. пути прохождения тока через тело человека (наибольшая опасность возникает при непосредственном прохождении тока через жизненно важные органы: сердце, легкие, головной мозг);
4. продолжительности воздействия на организм человека (с течением времени резко падает сопротивление кожи человека, более вероятным становится поражение сердца, и накапливаются другие отрицательные последствия);
5. условий внешней среды.

Согласно ГОСТ 12.1.038-82, человек начинает ощущать протекающий через него ток в 0,3 мА (50 Гц), 0,4 мА (400 Гц) и 1 мА (постоянный). Это пороговый ощутимый ток. Ток 10 – 15 мА (50 Гц) называется пороговым не отпускающим. Он вызывает судорожные сокращения мышц руки, в которой зажат проводник. Ток 25 – 50 мА (50 Гц) приводит к затруднению и даже прекращению дыхания, а при 100 мА ток вызывает остановку или фибрилляцию сердца (хаотические и разновременные сокращения волокон сердечной мышцы, полностью нарушающие ее работу как насоса), прекращению кровообращения и смерть. При постоянном токе пороговый не отпускающий ток 50 – 70 мА, а фибрилляционный – до 0,3 А.

* + - 1. Электромагнитное и ионизирующее излучение

Электромагнитным излучением называется излучение, прямо или косвенно вызывающее ионизацию среды. Контакт с электромагнитными излучениями представляет серьезную опасность для человека.

Основным источником электромагнитного излучения при работе с ПЭВМ является монитор. Дисплей излучает электромагнитные поля (ЭМП) в очень ши­роком диапазоне частот (от 3 Гц до 300 мГц), но преобладают следующие два диапазона:

1. поля, создаваемые блоком сетевого питания и блоком кадровой развертки дисплея (например, с частотой 50–150 Гц – электромагнитные поля от блока питания, проводов и системы вертикального отклонения и модуляции луча ЭЛТ); основной энергетический спектр этих полей сосредоточен в диапазоне частот до 1 кГц;
2. поля, создаваемые блоком строчной развертки и блоком сетевого питания ПЭВМ (если он импульсный); основной энергетический спектр этих полей сосредоточен в диапазоне частот от 15 до 100 кГц.

Защита от электромагнитного излучения компьютера:

1. по возможности, стоит приобрести жидкокристаллический монитор, поскольку его излучение значительно меньше, чем у распространённых ЭЛТ мониторов (монитор с электроннолучевой трубкой);
2. системный блок и монитор должен находиться как можно дальше от человека;
3. не оставлять компьютер включённым на длительное время, если он не используется, например, использовать "спящий режим" для монитора;
4. в связи с тем, что электромагнитное излучение от стенок монитора намного больше, лучше постараться поставить монитор в угол, так что бы излучение поглощалось стенами. Особое внимание стоит обратить на расстановку мониторов в офисах;
5. по возможности сократить время работы за компьютером и чаще прерывать работу;
6. компьютер должен быть заземлён. Если есть защитный экран, то его тоже следует заземлить, для этого специально предусмотрен провод на конце которого находиться металлическая прищепка (не цепляйте её к системному блоку).

Ионизирующее излучение – это любое излучение, вызывающее ионизацию среды, т.е. протекание электрических токов в этой среде, в том числе и в организме человека, что часто приводит к разрушению клеток, изменению состава крови, ожогам и другим тяжелым последствиям.

Излучения на расстоянии 40 см от экрана составляют около 0.08 мкР/ч, что не превышает нормы. И по данному фактору можно отнести работы с персональным компьютером к допустимым по степени вредности.

Исходя из вышесказанного, условия работы с персональным компьютером удовлетворяют требованиям Р 2.2.013-94 и СанПиН 2.2.2./2.4.1340‑03, но необходимы дополнительные меры защиты в виде регламентирования рабочего времени.

* + - 1. Освещенность

Правильно спроектированное и выполненное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на работающего, повышает безопасность труда и снижает травматизм.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Существует три вида освещения - естественное, искусственное и совмещенное (естественное и искусственное вместе).

Естественное освещение - освещение помещений дневным светом, проникающим через световые проемы в наружных ограждающих конструкциях помещений.

Естественное освещение характеризуется тем, что меняется в широких пределах в зависимости от времени дня, времени года, характера области и ряда других факторов.

Искусственное освещение применяется при работе в темное время суток и днем, когда не удается обеспечить нормированные значения коэффициента естественного освещения (пасмурная погода, короткий световой день).

Освещение, при котором недостаточное по нормам естественное освещение дополняется искусственным, называется совмещенным освещением.

Искусственное освещение подразделяется на рабочее, аварийное, эвакуационное, охранное. Рабочее освещение, в свою очередь, может быть общим или комбинированным. Общее - освещение, при котором светильники размещаются в верхней зоне помещения равномерно или применительно к расположению оборудования. Комбинированное - освещение, при котором к общему добавляется местное освещение.

Согласно СНиП II-4-79 в помещениях вычислительных центров необходимо применить систему комбинированного освещения.

При выполнении работ категории высокой зрительной точности (наименьший размер объекта различения 0,3 - 0,5 мм) величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5%, а при зрительной работе средней точности (наименьший размер объекта различения 0,5 - 1,0 мм) КЕО должен быть не ниже 1,0%. В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, которые должны располагаться над рабочими поверхностями равномерно.

Требования к освещенности в помещениях, где установлены компьютеры, следующие: при выполнении зрительных работ высокой точности общая освещенность должна составлять 300лк, а комбинированная - 750лк; аналогичные требования при выполнении работ средней точности - 200 и 300лк соответственно.

Кроме того, все поле зрения должно быть освещено достаточно равномерно – это основное гигиеническое требование. Иными словами, степень освещения помещения и яркость экрана компьютера должны быть примерно одинаковыми, т.к. яркий свет в районе периферийного зрения значительно увеличивает напряженность глаз и, как следствие, приводит к их быстрой утомляемости.

* + 1. Анализ воздействия на окружающую среду

Под опасным воздействием на окружающую среду понимается воздействие при определенных условиях (случайного или детерминированного характера) на элементы окружающей среды, приводящее к одному или к совокупности следующих нежелательных последствий: ухудшению здоровья человека по сравнению со среднестатистическим значением, т. е. приводящее к заболеванию или даже к смерти человека; ухудшению состояния окружающей человека среды, обусловленное нанесением материального или социального ущерба (нарушением процесса нормальной хозяйственной деятельности, потерей того или иного вида собственности и т. д.) и/или ухудшением качества природной среды (ГОСТ Р 14.03–2005).

Опасные воздействия на окружающую среду обусловлены тем, что сырье, используемое в сборке компьютеров, является токсичным. При создании одного среднестатистического персонального компьютера общий вес различных химикатов и ископаемого топлива в 10 раз превышает вес окончательного продукта. Также электромагнитные поля, излучаемые экранами мониторов, разрушают ионную структуру воздуха. Это объясняется притяжением отрицательных ионов к экрану дисплея, находящегося под положительным потенциалом, и отталкиванием положительных.

* + 1. Анализ возможных чрезвычайных ситуаций

Чрезвычайная ситуация (ЧС) – это обстановка на определенной территории, сложившаяся в результате аварии, опасного природного явления, катастрофы, стихийного или иного бедствия, которые могут повлечь или повлекли за собой человеческие жертвы, ущерб здоровью людей или окружающей природной среде, значительные материальные потери и нарушение условий жизнедеятельности людей.

ЧС являются многофакторными событиями, которые могут возникать в результате многочисленных причин, в различных условиях и приводить к разнообразным последствиям.

По происхождению ЧС подразделяются на природные, техногенные, антропогенные, военные.

Под техногенной ЧС понимается состояние, при котором в результате возникновения источника техногенной ЧС на объекте, определенной территории или акватории нарушаются нормальные условия жизни и деятельности людей, возникает угроза их жизни и здоровью, наносится ущерб имуществу населения, народному хозяйству и окружающей среде (ГОСТ 22.0.05-94).

Авария – опасное техногенное происшествие, создающее на объекте, определенной территории или акватории угрозу жизни и здоровью людей и приводящее к разрушению зданий, сооружений, оборудования и транспортных средств, нарушению транспортного или производственного процесса, а также нанесению ущерба окружающей природной среде (ГОСТ 22.0.05-94). Крупная авария, как правило с человеческими жертвами, является катастрофой.

В соответствии с Постановлением Правительства РФ «О классификации чрезвычайных ситуаций природного и техногенного характера» (2007) ЧС подразделяются в зависимости от показателей:

1. количество людей, пострадавших в ЧС;
2. количество людей, у которых оказались нарушены условия жизнедеятельности;
3. размер материального ущерба;
4. размер зоны распространения поражающих факторов.

При идентификации возможных техногенных ЧС, связанных с объектом проектирования, необходимо провести их анализ в зависимости от происхождения, масштаба распространения, вида поражающих факторов. Так, например, для котельной возможными ЧС являются пожар, взрыв, вызванные воспламенением газа, мазута; разгерметизация систем, работающих под давлением, и воздействие рабочих сред на человека, а аварии в системах электроснабжения приведут к потере их устойчивости.

Существует ряд отраслей производства, которые, в случае возникновения на них аварий, могут создавать наиболее опасные ситуации. Они относятся к опасным производственным объектам.

Из анализа промышленных аварий и катастроф следует, что причинами ЧС зачастую являются ошибки при проектировании и недостаточный уровень современных знаний.

Анализ потенциально опасных факторов, связанных с проектируемым объектом, должен явиться основой для обоснования необходимости расчета защиты от наиболее опасного фактора.

К техногенным относят ЧС, происхождение которых свя­зано с техническими объектами, — пожары, взрывы, аварии на химически опасных объектах, выбросы радиоактивных веществ, обрушение зданий, аварии на системах жизнеобеспечения.

К природным относятся ЧС, связанные с проявлением сти­хийных сил природы, — землетрясения, наводнения, изверже­ния вулканов, оползни, сели, ураганы, смерчи, бури, природные пожары и др.

К экологическим ЧС относятся аномальное природное загрязнение атмосферы, разрушение озонового слоя земли, опустынивание земель, засоление почв, кислотные дожди и др.

К биологическим ЧС относятся эпидемии, эпизоотии, эпифитотии.

К социальным ЧС относятся события, происходящие в об­ществе, — межнациональные конфликты, терроризм, грабежи, геноцид, войны и др.

Антропогенные ЧС являются следствием ошибочных дейс­твий людей.

Чрезвычайные ситуации классифицируются в зависимости от количества людей, пострадавших в этих ситуациях, людей, у которых оказались нарушены условия жизнедеятельности, от размера материального ущерба, а также границы зон распространения поражающих факторов чрезвычайной ситуации.

Анализ чрезвычайных ситуаций, имевших место в России за последние годы, позволил выделить причины аварийности и травматизма:

1. человеческий фактор — 50,1%;
2. оборудование, техника — 18,1%;
3. технология выполнения работ — 7,8%;
4. условия внешней среды — 16,6%;
5. прочие факторы — 7,4%.

В настоящее время заметно возрос удельный вес аварий, происходящих из-за неправиль­ных действий обслуживающего технического персонала (более 50%). Часто это связано с недостаточностью профессионализма, а также неумением принимать оптимальные решения в сложной критической обстановке в условиях дефицита времени.

* 1. Мероприятия по охране труда

Охрана труда – это система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия.

Условно охрану труда (ОТ) можно представить совокупностью четырех составляющих:

1. правовая охрана труда (ПОТ);
2. техника безопасности (ТБ);
3. производственная санитария (ПС);
4. пожарная безопасность (ПБ).

В соответствии со ст. 210 ТК РФ основными направлениями государственной политики в области охраны труда являются:

1. обеспечение приоритета сохранения жизни и здоровья работников;
2. принятие и реализация федеральных законов и иных нормативных правовых актов Российской Федерации, законов и иных нормативных правовых актов субъектов Российской Федерации в области охраны труда, а также федеральных целевых, ведомственных целевых и территориальных целевых программ улучшения условий и охраны труда;
3. государственное управление охраной труда;
4. государственный надзор и контроль за соблюдением государственных нормативных требований охраны труда;
5. государственная экспертиза условий труда;
6. установление порядка проведения аттестации рабочих мест по условиям труда и порядка подтверждения соответствия организации работ по охране труда государственным нормативным требованиям охраны труда;
7. содействие общественному контролю за соблюдением прав и законных интересов работников в области охраны труда;
8. профилактика несчастных случаев и повреждения здоровья работников;
9. расследование и учет несчастных случаев на производстве и профессиональных заболеваний;
10. защита законных интересов работников, пострадавших от несчастных случаев на производстве и профессиональных заболеваний, а также членов их семей, на основе обязательного социального страхования работников от несчастных случаев на производстве и профессиональных заболеваний;
11. установление компенсаций за тяжелую работу и работу с вредными и (или) опасными условиями труда;
12. координация деятельности в области охраны труда, охраны окружающей природной среды и других видов экономической и социальной деятельности;
13. распространение передового отечественного и зарубежного опыта работы по улучшению условий и охраны труда;
14. участие государства в финансировании мероприятий по охране труда;
15. подготовка специалистов по охране труда и повышение их квалификации;
16. организация государственной статистической отчетности об условиях труда, а также о производственном травматизме, профессиональной заболеваемости и об их материальных последствиях;
17. обеспечение функционирования единой информационной системы охраны труда;
18. международное сотрудничество в области охраны труда;
19. проведение эффективной налоговой политики, стимулирующей создание безопасных условий труда, разработку и внедрение безопасных техники и технологий, производство средств индивидуальной и коллективной защиты работников;
20. установление порядка обеспечения работников средствами индивидуальной и коллективной защиты, а также санитарно-бытовыми помещениями и устройствами, лечебно-профилактическими средствами.

Производственные процессы должны быть пожаро- и взрывобезопасными, а также не должны загрязнять окружающую среду (воздух, почву, водоемы) выбросами вредных веществ.

* + 1. Мероприятия по обеспечению комфортных условий труда

 В целях предотвращения неблагоприятного влияния на здоровье работников вредных факторов производственной среды и трудового процесса при использовании ими персональных электронно-вычислительных машин (ПЭВМ) режим их работы рекомендовано устанавливать в зависимости от вида и категории трудовой деятельности.

Виды трудовой деятельности разделяются на три группы: группа А - работа по считыванию информации с монитора компьютера с предварительным запросом; группа Б - работа по вводу информации; группа В - творческая работа в режиме диалога с ПЭВМ. При выполнении в течение рабочей смены работ, относящихся к разным видам трудовой деятельности, за основную работу с ПЭВМ следует принимать такую, которая занимает не менее 50% времени в течение рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливаются три категории тяжести и напряженности работы с ПЭВМ, которые определяются: для группы А - по суммарному числу считываемых знаков за рабочую смену, но не более 60000 знаков за смену; для группы Б - по суммарному числу считываемых или вводимых знаков за рабочую смену, но не более 40000 знаков за смену; для группы В - по суммарному времени непосредственной работы с ПЭВМ за рабочую смену, но не более 6 часов за смену.

В зависимости от категории трудовой деятельности и уровня нагрузки за рабочую смену при работе с ПЭВМ устанавливается суммарное время регламентированных перерывов.

Для предупреждения преждевременной утомляемости пользователей ПЭВМ рекомендуется организовывать рабочую смену путем чередования работ с использованием ПЭВМ и без него.

При возникновении у работающих с ПЭВМ зрительного дискомфорта и других неблагоприятных субъективных ощущений, несмотря на соблюдение санитарно-гигиенических и эргономических требований, рекомендуется применять индивидуальный подход с ограничением времени работы с ПЭВМ.

В случаях, когда характер работы требует постоянного взаимодействия с монитором компьютера (набор текстов или ввод данных и т.п.) с напряжением внимания и сосредоточенности, при исключении возможности периодического переключения на другие виды трудовой деятельности, не связанные с ПЭВМ, рекомендуется организация перерывов на 10 - 15 мин. через каждые 45 - 60 мин. работы.

Продолжительность непрерывной работы с ПЭВМ без регламентированного перерыва не должна превышать одного часа.

При работе с ПЭВМ в ночную смену (с 22 до 6 часов) независимо от категории и вида трудовой деятельности продолжительность регламентированных перерывов следует увеличивать на 30%.

Существует множество превентивных (предупредительных) мероприятий, позволяющих повысить безопасность работы. Одна из них заключается в создании на рабочем месте соответствующего инженерного обеспечения. Задача – сделать работу более комфортабельной, менее утомительной, помочь работнику стать более бдительным, менее открытым для несчастных случаев.

Работающим на ПЭВМ с высоким уровнем напряженности во время регламентированных перерывов и в конце рабочего дня рекомендуется посещать специально оборудованные комнаты для снятия напряжения.

* + 1. Мероприятия по защите от опасных и вредных производственных факторов

Задачей защиты человека от опасных вредных производственных факторов (ОВПФ) является снижение уровня вредных факторов, не превышающих ПДУ и ПДК и риска появления опасных факторов до величин приемлемого риска.

Основные мероприятия по защите человека от опасных и вредных производственных факторов приведены ниже:

1. совершенствование технологии производств и технических средств с целью снижения уровня ОВПФ;
2. защита расстоянием (удаление от источника ОВПФ);
3. защита временем (уменьшение времени пребывания в зоне действия ОВПФ);
4. применение средств защиты:
   * применение средств коллективной защиты;
   * применение средств индивидуальной защиты.

Защита человека от физических негативных факторов осуществляется тремя основными методами:

1. ограничение времени пребывания в зоне действия физического поля;
2. удаление от источника поля;
3. применение средств защиты.

Для защиты от акустических колебаний (шума, ультра и инфразвука) проводят следующие мероприятия:

1. снижение звуковой мощности источника звука;
2. размещение рабочих мест с учетом направленности излучения от источника звука;
3. акустическая обработка помещений (применение звукопоглощения облицовки, штучные, объемные поглотители различных конструкций, подвешенные к потолку помещений);
4. применение звукоизоляции (глушители);
5. применение средств индивидуальной защиты (наушники, шлемы, беруши).

Для снижения воздействия электромагнитного и ионизирующего излучения рекомендуется применять мониторы с пониженным уровнем излучения, устанавливать защитные экраны, а также соблюдать регламентированные режимы труда и отдыха.

Защита работника от негативного воздействия источника внешнего ионизирующего излучения достигается путем:

1. снижение мощности источника излучения до минимально необходимой величины;
2. увеличение расстояния между источником излучения и работником;
3. уменьшение продолжительности работы в зоне излучения;
4. установление между источником излучения и работником защитного.
   1. Мероприятия по охране окружающей среды

Охраной окружающей среды называется комплекс мер, направленных на предупреждение отрицательного влияния человеческой деятельности на природу, обеспечение благоприятных и безопасных условий жизнедеятельности человека. В условиях научно-технического прогресса важнейшей задачей человечества является охрана важнейших элементов окружающей среды (воздух, вода, почва), которые из-за вредных промышленных выбросов и отходов подвергаются сильнейшему загрязнению. Результатом чего является закисление почвы и воды, изменение климата и разрушение озонового слоя. Именно поэтому охране окружающей среды в строительстве отводится важное место в общегосударственных задачах. В последние годы, в связи с необратимыми процессами и изменениями окружающей среды, вопросы охраны среды выросли в общемировую проблему. Поэтому разработка долгосрочной экологической политики по созданию благоприятных условий (пдв) стала необходима.

Для охраны окружающей среды необходимо разработать и освоить оптимальную технологию утилизации устаревших или пришедших в негодность комплектующих ПЭВМ. Оставшиеся исправные детали можно использовать для ремонтной организации или служить запасными частями. Для неисправных деталей необходима соответствующая утилизация, которую будет предусматривать безопасную деятельность для окружающий среды.

* 1. Мероприятия по защите от чрезвычайных ситуаций

В качестве основных направлений в решении задач обеспечения защиты от чрезвычайных ситуаций могут рассматриваться следующие:

1. прогнозирование и оценка возможных последствий чрезвычайных ситуаций;
2. планирование мероприятий по предотвращению или уменьшению вероятности возникновения чрезвычайных ситуаций, а также сокращению масштабов их последствий;
3. обеспечение устойчивой работы объектов народного хозяйства в чрезвычайных ситуациях;
4. обучение населения действиям в чрезвычайных ситуациях;
5. ликвидация последствий чрезвычайных ситуаций.

Для тушения пожаров в рассматриваемом помещении нужно использовать либо порошковые составы, либо установки углекислотного тушения, т.к. при использовании воды и пены велика вероятность поражения электрическим током.

Выбор типа огнетушителя (передвижной или ручной) обусловлен размерами возможных очагов пожара. При их значительных размерах необходимо использовать передвижные огнетушители.

Число огнетушителей одного из типов для разных категорий помещений необходимо устанавливать из таблиц, приведенных в нормах оснащения помещений ручными или передвижными огнетушителями.

Расстояние от возможного очага пожара до места размещения огнетушителя не должно превышать 30 м для помещений категории В и 70 м. для помещений категории Д.

Огнетушители должны всегда содержаться в исправном состоянии, периодически осматриваться, проверяться и своевременно перезаряжаться.

Рассматриваемое рабочее место оборудовано огнетушителем и системой пожарной сигнализации.

Дополнительных мер по защите от ЧС не требуется.

* 1. Расчетная часть

В данном подразделе приведены необходимые расчёты:

* освещённости;
* уровня шума.
  + 1. Расчет освещенности

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ:

1. по спектральному составу света они близки к дневному, естественному свету;
2. обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);
3. обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
4. более длительный срок службы.

Расчет освещения производится для комнаты площадью 16 м2, длина которой 4 м, ширина - 4 м. Воспользуемся методом светового потока.

Для определения количества светильников определим световой поток, падающий на поверхность по формуле 8.1:

, (8.1)

где F - рассчитываемый световой поток, Лм;

Е - нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет Е = 200Лк;

S - площадь освещаемого помещения (в нашем случае S = 16 м2);

Z - отношение средней освещенности к минимальной (обычно принимается равным 1,1…1,2, пусть Z = 1,1);

К - коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (К = 1,2);

n - коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы), значение коэффициентов РС и РП: РС = 40%, РП = 60%. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле 8.2:

, (8.2)

где S - площадь помещения, S = 16 м2;

h - расчетная высота подвеса, h = 2.8 м;

A - ширина помещения, А = 4 м;

В - длина помещения, В = 4 м.

Подставив значения получим:



Зная индекс помещения I, по таблице находим n = 0,28. Подставим все значения в формулу для определения светового потока F:



Для освещения выбираем люминесцентные лампы типа T8 StandartSpecialLengthF30W/54-765, световой поток которых F = 4450 Лк.

Для освещения выбираем люминесцентные лампы типа OSRAM HO 54 W/830 G5, световой поток которых F = 4450 Лк.

Рассчитаем необходимое количество ламп по формуле 8.3:

, (8.3)

где N - определяемое число ламп;

F - световой поток, F = 15058 Лм;

Fл - световой поток лампы, Fл = 4450 Лм.



При выборе осветительных приборов используем светильники типа ОД.

* + 1. Расчет уровня шума

Одним из неблагоприятных факторов производственной среды является вы­со­кий уровень шума, создаваемый печатными устройствами, оборудованием для кон­ди­ци­онирования воздуха, вентиляторами систем охлаждения в самих ПЭВМ.

Для решения вопросов о необходимости и целесообразности снижения шума не­обхо­димо знать уровни шума на рабочем месте оператора.

Уровень шума, возникающий от нескольких некогерентных источников, работа­ющих одновременно, подсчитывается на основании принципа энергетического сумми­рования излучений отдельных источников по формуле 8.4:

, (8.4)

где *Li* – уровень звукового давления i-го источника шума;

n – количество источников шума.

Полученные результаты расчета сравнивается с допустимым значением уровня шу­ма для данного рабочего места. Если результаты расчета выше допустимого значения уров­ня шума, то необходимы специальные меры по снижению шума. К ним отно­сятся: обли­цовка стен и потолка зала звукопоглощающими материалами, снижение шума в источ­нике, правильная планировка оборудования и рациональная организация рабочего места оператора.

Уровни звукового давления источников шума, действующих на оператора на его ра­бочем месте представлены в таблице 8.2.

Таблица 8.2 - Уровни звукового давления различных источников

|  |  |
| --- | --- |
| Источник шума | Уровень шума, дБ |
| Жесткий диск | 40 |
| Вентилятор | 45 |
| Монитор | 17 |
| Клавиатура | 10 |
| Принтер | 45 |
| Сканер | 42 |

Обычно рабочее место оператора оснащено следующим оборудованием: винчестер в системном блоке, вентилятор(ы) систем охлаждения ПЭВМ, монитор, клавиатура, прин­тер и сканер. Подставив значения уровня звукового давления для каждого вида оборудования в формулу, получим:

L∑ = 10 · lg (104+104,5+101,7+101+104,5+104,2) = 49,6 дБ

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ (ГОСТ 12.1.003-83). И если учесть, что вряд ли такие перифе­рийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже.

* 1. Оценка эффективности

В данном подразделе был произведен анализ основных вредных и опасных факторов исследуемого объекта. В ходе исследований был проведен выбор си­стемы и расчет оптимального освещения производственного помещения, а также расчет уровня шума на рабочем месте. Соблюдение условий, определяющих оптимальную ор­ганизацию рабочего места бакалавра, позволит сох­ранить хорошую ра­ботоспособность в течение всего рабочего дня, повысит как в ко­личественном, так и в качественном отношениях производительность труда програм­миста, что в свою очередь будет способствовать быстрейшей разработке и отладке программного продукта.

# Заключение

В результате дипломного проектирования была разработана одна из подсистем ОУ - "Портал управляющих компаний" для АИС:Объектовый учёт. Все поставленные задачи были успешно выполнены.

В рамках выпускной квалификационной работы были выполнены следующие задачи:

1. Раскрытие информации об организациях, осуществляющих деятельность в сфере управления многоквартирными домами в рамках Постановления Правительства Российской Федерации от 23 сентября 2010 г. N 731 г. Москва "Об утверждении стандарта раскрытия информации организациями, осуществляющими деятельность в сфере управления многоквартирными домами";
2. Предоставление сведений:
   1. Об оказываемых услугах по содержанию и ремонту жилья, порядке и условиях оказания;
   2. О планируемых, проведенных работах и их стоимости;
   3. О тарифах, нормативах на коммунальные услуги;
   4. О начислениях, перерасчётах, показаниях приборов учёта, сборах, задолженности по дому;
   5. По участию дома в программе капитального ремонта.

Основываясь на структуре разработанной ИС, можно выделить следующие функции объекта автоматизации:

1. Формирование реестра региональных сегментов ГИС ЖКХ;
2. Выявление недобросовестных компаний;
3. Раскрытие и предоставление информации.

# Список использованных источников

1. Албахари, Д. C# 5.0. Справочник. Полное описание языка / Д. Албахари. - 1-е изд. - М.: Вильямс, 2013. – 1008 с.
2. Бен-Ган, И. Microsoft SQL Server 2012. Высокопроизводительный код T-SQL. Оконные функции / И. Бен-Ган. - М.: Русская Редакция, 2013. – 256 с.
3. Гончар, С. Т. Безопасность и экологичность объекта проектирования: учебное пособие по дипломному проектированию / С. Т. Гончар. – 2-е изд. – Ульяновск: УлГТУ, 2009. – 165 с.
4. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.– М. : Стандартинформ, 2010.– 8 c.
5. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М. : Стандартинформ, 1996.– 9 c.
6. ГОСТ 34.601-90. Автоматизированные системы. Стадии создания. – М. : Изд-во стандартов, 1997.–5 c.
7. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. М. : Стандартинформ, 2010.– 166.
8. Лобел. Л. Разработка приложений на основе Microsoft SQL Server 2008 / Л. Лобел. - М.: Русская Редакция, 2010. – 1024 с.
9. Родионов, В. В. Дипломное проектирование: учебно-методическое пособие для студентов специальности 23020165 «Информационные системы и технологии» / В. В. Родионов. – Ульяновск: УлГТУ, 2008. – 98 с.
10. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4 / Э. Троелсен. – 5-е изд. – М. : Вильямс, 2010. – 1392 с.
11. Шилдт, Г. C# 4.0. Полное руководство / Г. Шилдт. - 1-е изд. - М.: Вильямс, 2013. - 1056 с.
12. Availability of Features in Visual Studio Versions // Microsoft Developer Network. – [Б. м. : б. и.], 2015. – Режимдоступа: https://msdn.microsoft.com/enus/library/ee519072.aspx (датаобращения: 15.05.2015)

# Приложение А

(обязательное)

**Текст программы**

Программный код модуля Site

Файл ContentController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Areas.Site.Models;

using HM\_web\_4\_people.Models;

namespace HM\_web\_4\_people.Areas.Site.Controllers

{

public class ContentController : MinisiteController

{

ArticleRepository Repository;

public ContentController()

: this(null) { }

public ContentController(ArticleRepository Repository)

{

this.Repository = Repository??new ArticleRepository();

}

public ActionResult Index()

{

return Error;

}

public ActionResult News(int? id, int? page)

{

if (id.HasValue) return Article(id.Value, page);

return Category(RouteData.Values["action"].ToString(), page);

}

public ActionResult Announcements(int? id, int? page)

{

if (id.HasValue) return Article(id.Value, page);

return Category(RouteData.Values["action"].ToString(), page);

}

public ActionResult LegalCulture(int? id, int? page)

{

if (id.HasValue) return Article(id.Value, page);

return Category(RouteData.Values["action"].ToString(), page);

}

public ActionResult Category(string id, int? page)

{

if (Organization == null) return Error;

var Category = Repository.GetCategory(id);

if (Category == null) return Error;

ViewBag.Category = Category;

ViewBag.Page = page;

return View("Category", new PaginatedList<vw\_Article> (Repository.GetArticleList(Organization.OrgId, Category.Id), page ?? 0));

}

public ActionResult Article(int id, int? page)

{

if (Organization == null) return Error;

var Item = Repository.Get(Organization.OrgId, id);

if (Item == null || Item.IsPage) return Error;

ViewBag.Category = Repository.GetCategory(Item.CategoryId);

ViewBag.Page = page;

return View("Article", Item);

}

public ActionResult Page(string id)

{

if (Organization == null) return Error;

var Item = Repository.GetPage(Organization.OrgId, id);

return View("Page", Item);

}

}

}

Файл ContractController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Areas.Site.Models;

using HM\_web\_4\_people.Models;

using AIS.HM.Model;

namespace HM\_web\_4\_people.Areas.Site.Controllers

{

public class ContractController : MinisiteController

{

ContractRepository Repository;

public ContractController()

: this(null) { }

public ContractController(ContractRepository Repository)

{

this.Repository = Repository ?? new ContractRepository();

}

public ActionResult Index()

{

return Error;

}

public ActionResult EntranceProtection(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult ParkingProtection(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult CommonPlaceOnLease(int? page)

{

if (Organization == null) return Error;

var list = (new DB()).vw\_od\_TotalEquityAssetContract.Where

(x => x.ContractorId == Organization.OrgId).OrderBy(x => x.Date);

return View(new PaginatedList<vw\_od\_TotalEquityAssetContract>(list, page ?? 0));

}

public ActionResult Other(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult CommunalResources(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult Contractor(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult EnergyAudit(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult EnergyService(int? id, int? page)

{

return Type(RouteData.Values["action"].ToString(), page);

}

public ActionResult Type(string id, int? page)

{

if (Organization == null) return Error;

var Type = Repository.GetType(id);

if (Type == null) return Error;

ViewBag.Type = Type;

ViewBag.Page = page;

return View("Type", new PaginatedList<vw\_Contract> (Repository.GetList(Organization.OrgId, Type.Id, DateTime.Today), page ?? 0));

}

}

}

Файл FinancialReportController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Areas.Site.Models;

using HM\_web\_4\_people.Models;

using AIS.HM.Model;

namespace HM\_web\_4\_people.Areas.Site.Controllers

{

public class FinancialReportController : MinisiteController

{

private DB db = new DB();

public ActionResult Index(int? id)

{

vw\_cmn\_OrganizationFinancialReport item = null;

if (Organization == null) return Error;

var list = db.vw\_cmn\_OrganizationFinancialReport.Where(d => d.OrganizationId == Organization.OrgId).OrderByDescending(d => d.DateStart);

if (list.Any())

{

if (id.HasValue)

item = list.FirstOrDefault(d => d.Id == id);

else

item = list.FirstOrDefault();

}

else

return View(new FinancialReportViewModel(null, null));

if (item == null) return Error;

SelectList periodOptions = new SelectList(list.ToList().Select(d => new { Value = d.Id, Text = (d.DateStart.Month == d.DateEnd.AddMonths(-1).Month && d.DateStart.Year == d.DateEnd.AddMonths(-1).Year ? d.DateStart.ToString("MMMM yyyy") : d.DateStart.ToString("MMMM yyyy") + " - " + d.DateEnd.AddMonths(-1).ToString("MMMM yyyy"))

}), "Value", "Text", item.Id);

return View(new FinancialReportViewModel(item, periodOptions));

}

}

}

Файл IndexController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Areas.Site.Models;

using HM\_web\_4\_people.Models;

using Helpers;

namespace HM\_web\_4\_people.Areas.Site.Controllers

{

public class IndexController : MinisiteController

{

OrganizationInformationRepository Repository;

public IndexController()

: this(new OrganizationInformationRepository())

{ }

public IndexController(OrganizationInformationRepository Repository)

{

this.Repository = Repository;

}

// GET: /Site/Index/

public ActionResult Index()

{

if (Organization == null) return Error;

ArticleRepository newsRep = new ArticleRepository();

var licenses = new DBClassesDataContext().vw\_OrganizationLicenses.Where(d => d.OrganizationId == Organization.OrgId).OrderByDescending(d => d.LicenseDate);

return View(new OrganizationInformationViewModel(

Organization,

licenses,

newsRep.GetArticleList(Organization.OrgId, 127).Take(Helpers.ConfigurationHelper.GetDefault("people.CountNewsShow", 3).Value),

Repository.GetAssociationList(Organization.OrgId)

));

}

public ActionResult Map()

{

if (Organization == null) return Error;

return View(new OrganizationInformationViewModel(

Organization,

Repository.GetAssociationList(Organization.OrgId)

));

}

[Authorize]

public ActionResult Message()

{

if (Organization == null) return Error;

MessageModel Model = new MessageModel();

Model.Organization = Organization;

if (string.IsNullOrWhiteSpace(Organization.Email)) return View("Message\_Cant", Model);

return View(Model);

}

[Authorize]

[HttpPost]

[CaptchaValidator]

public ActionResult Message(MessageModel Model, bool captchaValid, string captchaErrorMessage)

{

if (Organization == null) return Error;

Model.Organization = Organization;

if (!captchaValid)

ModelState.AddModelError("recaptcha\_response\_field", captchaErrorMessage);

try

{

if (!ModelState.IsValid) throw new Exception();

string Message = "Автор: "+Model.Name+"\n"+

"Контактная информация: "+Model.Contacts+"\n"+

"Сообщение:\n"+Model.Message;

EmailHelper.SendEmail(Organization.Email, "СообщениеспорталаУК", Message);

return View("Message\_Sent");

}

catch (Exception)

{

ModelState.AddModelError("", "Ошибка при отправке сообщения.");

}

return View(Model);

}

}

}

Файл ManagementContractController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using AIS.HM.Model;

using HM\_web\_4\_people.Areas.Site.Models;

using HM\_web\_4\_people.Models;

namespace HM\_web\_4\_people.Areas.Site.Controllers

{

public class ManagementContractController : MinisiteController

{

ManagementContractRepository Repository;

long FileGroupId { get { return 110; } }

public ManagementContractController()

: this(new ManagementContractRepository())

{ }

public ManagementContractController(ManagementContractRepository Repository)

{

this.Repository = Repository;

}

// Shows Cancelled Contracts

public ActionResult Index(int? page)

{

if (Organization == null) return Error;

ViewBag.Organization = Organization;

return View("List", new PaginatedList<vw\_RateContract>(Repository.GetTerminatedList(Organization.OrgId, DateTime.Today.Year - 1), page ?? 0));

}

}

}

ПрограммныйкодмодуляMain

Файл AccountController.cs

using System;

using System.Collections.Generic;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using System.Security.Principal;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

using System.Web.Security;

using HM\_web\_4\_people.Models;

using Helpers;

namespace HM\_web\_4\_people.Controllers

{

[HandleError]

public class AccountController : Controller

{

public IHMAuthenticationService FormsService { get; set; }

public IHMMembershipService MembershipService { get; set; }

protected override void Initialize(RequestContext requestContext)

{

if (FormsService == null) { FormsService = new HMAuthenticationService(); }

if (MembershipService == null) { MembershipService = new HMMembershipService(); }

base.Initialize(requestContext);

}

[Authorize]

[RegistrationCheck]

public ActionResult Index()

{

ViewBag.PasswordLength = MembershipService.MinPasswordLength;

var user = MembershipService.GetUserInformation(((HMIdentity)User.Identity).Id);

var addresses = MyAddressRepositoryFactory.GetRepository().GetList(((HMIdentity)User.Identity).Id);

return View(new AccountIndexViewModel() { UserInformation = user, Addresses = addresses });

}

[Authorize]

[RegistrationCheck]

[HttpPost]

public ActionResult Index(bool? editInformation, bool? changePassword)

{

ViewBag.PasswordLength = MembershipService.MinPasswordLength;

var user = MembershipService.GetUserInformation(((HMIdentity)User.Identity).Id);

var addresses = MyAddressRepositoryFactory.GetRepository().GetList(((HMIdentity)User.Identity).Id);

var passwordModel = new ChangePasswordModel();

try

{

if (editInformation == true)

{

UpdateModel(user, "UserInformation", new string[] { "Login", "Name", "Email" });

if (ModelState.IsValid) {

int Status = MembershipService.UpdateUser(user);

if (Status == 0)

return RedirectToAction("Index");

else

ModelState.AddModelError("", AccountValidation.UpdateErrorCodeToString(Status));

}

}

if (changePassword == true)

{

if (TryUpdateModel(passwordModel, "PasswordModel", new string[] { "Old", "New", "Confirm" }))

{

if (ModelState.IsValid)

{

if (MembershipService.ChangePassword( ((HMIdentity)User.Identity).Id, passwordModel.Old, passwordModel.New))

return View("ChangePasswordSuccess");

else

ModelState.AddModelError("PasswordModel.Old", "Текущийпарольуказанневерно.");

}

}

else

{

ModelState.AddModelError("", "Паролинесовпадают.");

}

}

}

catch (Exception ex)

{

ModelState.AddModelError("", ex.Message);

}

return View(new AccountIndexViewModel() { UserInformation = user, PasswordModel = passwordModel, Addresses = addresses });

}

[RegistrationCheck]

public ActionResult LogOn(string returnUrl)

{

if (Request.IsAuthenticated)

{

if (!String.IsNullOrEmpty(returnUrl))

return Redirect(returnUrl);

else

return RedirectToAction("Index", "Home");

}

return View();

}

public ActionResult BarnaulLogOn()

{

return View();

}

[RegistrationCheck]

[HttpPost]

public ActionResult LogOn(LogOnModel model, string returnUrl)

{

if (ModelState.IsValid)

{

long id = 0;

int status = 0;

status = MembershipService.ValidateUser(model.Login, model.Password, ref id);

if (status == 0)

{

FormsService.SignIn(id, model.RememberMe);

if (!String.IsNullOrEmpty(returnUrl))

{

return Redirect(returnUrl);

}

else

{

return RedirectToAction("Index", "MyAddresses");

}

}

else

{

string Message;

switch (status)

{

case 99:

Message = "Пользователь заблокирован";

break;

default:

Message = "Логин или пароль указаны неверно";

break;

}

ModelState.AddModelError("", Message);

}

}

return View(model);

}

public ActionResult LogOff()

{

FormsService.SignOut();

return RedirectToAction("Index", "Home");

}

[RegistrationCheck]

public ActionResult Register()

{

ViewBag.PasswordLength = MembershipService.MinPasswordLength;

return View();

}

[HttpPost]

[RegistrationCheck]

[CaptchaValidator]

public ActionResult Register(RegisterModel model, bool captchaValid, string captchaErrorMessage)

{

if (!captchaValid)

ModelState.AddModelError("recaptcha\_response\_field", captchaErrorMessage);

if (ModelState.IsValid)

{

// Attempt to register the user

MembershipCreateStatus createStatus = MembershipService.CreateUser(model);

if (createStatus == MembershipCreateStatus.Success)

{

FormsService.SignIn(model.Id, false /\* createPersistentCookie \*/);

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError("", AccountValidation.ErrorCodeToString(createStatus));

}

}

// If we got this far, something failed, redisplay form

ViewBag.PasswordLength = MembershipService.MinPasswordLength;

return View(model);

}

[RegistrationCheck]

public ActionResult LostPassword()

{

return View();

}

[RegistrationCheck]

[HttpPost]

public ActionResult LostPassword(string Login)

{

// get user

if (MembershipService.RecoverPassword(Login))

{

return RedirectToAction("LostPasswordSuccess");

}

else

{

ModelState.AddModelError("", "");

}

return View(new LogOnModel

{

Login = Login

});

}

}

}

Файл BigRepairController.cs

using HM\_web\_4\_people.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Areas.Site.Models;

namespace HM\_web\_4\_people.Controllers

{

public class BigRepairController : Controller

{

DBClassesDataContext db = new DBClassesDataContext();

BigRepairRepository Repository;

protected override void Initialize(System.Web.Routing.RequestContext requestContext)

{

base.Initialize(requestContext);

if (Repository == null) Repository = new BigRepairRepository();

}

public ActionResult List(HouseFilterModel filter)

{

var list = Repository.GetRevisionsList();

var revisions = new RevisionCommissionViewModel(list, new RevisionCommissionFilter());

return View(new LongTermPlanViewModel(

filter,

Repository.GetList(),

revisions

));

}

public ActionResult Details(BigRepairDetailsFilter filter)

{

var item = Repository.Get(filter.id);

return View(DetectViewName(true), new LongTermPlanDetailsViewModel(filter, item, Repository.GetObjectList(item.Id)));

}

public ActionResult ObjectDetails(int oId, BigRepairDetailsFilter filter)

{

var plan = Repository.Get(filter.id);

var planObject = Repository.GetObject(filter.id, oId);

var rep = new Models.HouseInformationRepository();

int? lastBigRepairYear = rep.GetCommonByAddress(planObject.AdrId).LastBigRepairYear;

Dictionary<int, decimal?> elements = new Dictionary<int, decimal?>();

var elementvariants = from d in db.vw\_br\_MaxRate\_alts

where d.DateStart <= plan.Updated

&& (d.DateEnd == null || d.DateEnd > plan.Updated)

select new { d.ElementId, d.CostPerUnit };

List<int> element = elementvariants.Select(d => d.ElementId).Distinct().ToList();

foreach (var el in element)

elements.Add(el, elementvariants.Where(d => d.ElementId == el).Max(d => d.CostPerUnit));

ViewBag.Updated = plan.Updated;

ViewBag.filter = filter;

ViewBag.plan = plan;

ViewBag.item = planObject;

ViewBag.last = lastBigRepairYear;

ViewBag.elementvariants = elements;

ViewBag.rates = Repository.GetVariantList(planObject.Id).ToList();

ViewBag.income = Repository.GetIncome(planObject.ObjectId);

return View(DetectViewName(true));

}

}

}

Файл HomeController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using HM\_web\_4\_people.Models;

using HM\_web\_4\_people.Areas.Site.Models;

namespace HM\_web\_4\_people.Controllers

{

[HandleError]

public class HomeController : Controller

{

public ActionResult Index(bool? reset)

{

// Menu reset

if (reset.HasValue)

{

Helpers.MenuHelper.Reset();

Helpers.DomainHelper.Update();

return RedirectToAction("Index");

}

// Domain check

if (Helpers.DomainHelper.CheckOrganizationSite(Request.Url))

{

var orgId = Helpers.DomainHelper.GetOrganizationId(Request.Url);

using (var db = new DBClassesDataContext())

{

var item = db.vw\_OrganizationInformations.FirstOrDefault(

i => i.OrgId == orgId);

return RedirectToAction("Index", "Index", new { area = "Site", org\_url = item.UrlPath });

}

}

HomePageViewModel model = new HomePageViewModel();

OrganizationInformationRepository orgRep = new OrganizationInformationRepository();

var list = new List<SelectListItem>();

if (Helpers.AppHelper.GetConfig("HideOrganizationSelector", "0") != "1")

{

list = orgRep.GetOptions(0).ToList();

}

model.OrganizationOptions = list;

model.AddressViewModel = new AddressSelectorViewModel();

var \_db = new DBClassesDataContext();

if (Helpers.ConfigurationHelper.Get("people.ShowBRBlock", false, null) == true)

{

var acrList = from d in \_db.AccountChargedRaiseds

select d;

var types = Helpers.ConfigurationHelper.Get("people.AccountTypesForRaisedPercent", String.Empty, null);

if (!string.IsNullOrEmpty(types))

{

var typesArr = types.Trim().Split(',');

acrList = from d in acrList

join l in \_db.vw\_Accounts on d.AccountId equals l.Id

where typesArr.Contains(l.TypeCode)

select d;

}

//% собираемости по области

if (acrList.Any())

{

float raised = (float)acrList.Sum(m => m.Raised);

float charged = (float)acrList.Sum(m => m.Charged);

if (charged > 0)

model.PercentageRaised = 100 \* raised / charged;

else

model.PercentageRaised = 100;

}

else

model.PercentageRaised = 0;

//Кол-во объектов в программе КР

var \_longTermPlan = \_db.vw\_LongTermPlans.Where(m => m.IsPublished).OrderByDescending(m => m.Id).FirstOrDefault();

model.CountObjectLongTermPlan = (\_longTermPlan == null) ? -1 : \_db.vw\_LongTermPlanObjects.Count(m => m.PlanId == \_longTermPlan.Id);

//Кол-водомоввкраткосрочке

var \_shortTermPlan = \_db.vw\_br\_ShortTermPlans.Where(m =>

m.IsPublished).OrderByDescending(m => m.Id).FirstOrDefault();

model.CountObjectShortTermPlan = (\_shortTermPlan == null) ? -1 : \_db.vw\_br\_ShortTermPlanObjectYears.Where(m => m.PlanId == \_shortTermPlan.Id).Select(m => m.PlanObjectId).Distinct().Count();

}

// Кол-во УК, раскрывающих информацию на ПУК

model.CountUK = \_db.vw\_OrganizationInformations.Count(m => m.IsActive);

//Кол-вообъектовнаПУК

model.CountObject = \_db.vw\_Houses.Count();

var objectCountOver80 = \_db.vw\_Houses.Count(m => m.ObjectFillPart.HasValue && m.ObjectFillPart.Value >= (decimal)0.8 && m.ObjectTypeCode == "apartmentHouse");

model.PercentageObjectOver80 = (float)objectCountOver80 \* 100 / \_db.vw\_Houses.Count(m => m.ObjectTypeCode == "apartmentHouse"); ;

return View(model);

}

public ActionResult Message()

{

MessageSendModel model = new MessageSendModel();

if (Helpers.ConfigurationHelper.Get("people.SendMessageToAdmin", false, null) == true)

model.SolveToSendMessage = true;

else

model.SolveToSendMessage = false;

return View(model);

}

[HttpPost]

[CaptchaValidator]

public ActionResult Message(MessageSendModel model, bool captchaValid, string captchaErrorMessage)

{

if (!captchaValid)

ModelState.AddModelError("recaptcha\_response\_field", captchaErrorMessage);

if (Helpers.ConfigurationHelper.Get("people.SendMessageToAdmin", false, null) == true)

{

model.SolveToSendMessage = true;

try

{

if (!ModelState.IsValid) throw new Exception();

Helpers.EmailHelper.SendEmail("object@aisgorod.ru",

model.Theme,

string.Format("Пользователь: {0}\nКонтактыпользователя: {1}\nТекстсообщения: {2}", model.Name, model.Contact, model.Text)

);

model.MessageIsSend = true;

}

catch (Exception)

{

ModelState.AddModelError("", "Ошибка при отправке сообщения.");

return View(model);

}

}

else

ModelState.AddModelError("", "Ошибка при отправке сообщения.");

return View(model);

}

public ActionResult WhatToDo()

{

return View();

}

public ActionResult Stats()

{

var model = new StatsViewModel();

var list = new DBClassesDataContext().vw\_AccountChargedRaisedByRegions.OrderBy(i => i.RegionSort).ToList();

model.TotalList = list;

if (list.Any())

{

model.TopRegions = list.Where(i => !i.ParentId.HasValue).OrderByDescending(i => i.Percentage).Take(5);

model.WorseRegions = list.Where(i => !i.ParentId.HasValue).OrderBy(i => i.Percentage).Take(5);

}

return View(model);

}

}

}

Файл OrganizationController.cs

using HM\_web\_4\_people.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace HM\_web\_4\_people.Controllers

{

public class OrganizationController : Controller

{

private DBClassesDataContext db = new DBClassesDataContext();

// GET: /Organization/List

public ActionResult List(int? page)

{

var list = db.vw\_OrganizationInformations.OrderBy(i => i.ShortName);

var result = new PaginatedList<vw\_OrganizationInformation>(list, page);

return View(result);

}

}

}

Программныйкодмодуля BigRepair

Файл AccountController.cs

using Helpers;

using System;

using System.Collections.Generic;

using System.Data.Linq.SqlClient;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using WinterPreparations.Areas.BigRepair.Models;

using WinterPreparations.Models;

namespace WinterPreparations.Areas.BigRepair.Controllers

{

[Authorize]

public class AccountController : Controller

{

AccountRepository AccountRepository;

BankAccountRepository BankAccountRepository;

ObjectRepository ObjectRepository;

int objectTypeId;

OrganizationRepository OrganizationRepository;

protected override void Initialize(System.Web.Routing.RequestContext requestContext)

{

base.Initialize(requestContext);

AccountRepository = new AccountRepository();

BankAccountRepository = new BankAccountRepository();

ObjectRepository = new ObjectRepository();

objectTypeId = ObjectRepository.GetTypeId("apartmentHouse") ?? -1;

OrganizationRepository = new OrganizationRepository();

}

// Отображаетстраницуредактированиясчёта

// GET: /BigRepair/Account/AccountEdit/5

public ActionResult AccountEdit(long? id, int? bankAccountId)

{

vw\_Account item;

vw\_BankAccount bankItem = null;

if (!id.HasValue)

{

bankItem = BankAccountRepository.Get(bankAccountId.Value);

if (bankItem == null)

throw new ArgumentException();

item = new vw\_Account()

{

BankAccountId = bankAccountId.Value,

BankAccountName = bankItem.Name,

BankAccountNumber = bankItem.Number,

BankAccountStatusName = bankItem.StatusName,

BankAccountTypeName = bankItem.TypeName,

IncomeMonthStart = DateTime.Today,

DateStart = DateTime.Today

};

}

else

{

item = AccountRepository.Get(id.Value);

bankItem = BankAccountRepository.Get(item.BankAccountId);

}

if (item == null)

throw new ArgumentException();

var rate = new MinRateRepository().GetList().Where(d => (d.DateStart == null || d.DateStart <= DateTime.Today)

&& (d.DateEnd == null || d.DateEnd > DateTime.Today)).OrderBy(d => d.Rate).FirstOrDefault();

var decisionLink = oldDb.vw\_hm\_DecisionToObjectLinks.FirstOrDefault(m => m.Id == (item.DecisionLinkId ?? 0)) ?? new vw\_hm\_DecisionToObjectLink {

Rate = (rate == null)? 0 : rate.Rate,

RateDateStart = DateTime.Today

};

return View(new AccountViewModel(AccountRepository, item, bankItem, null, decisionLink));

}

// Редактируетсчёт

// POST: /BigRepair/Account/AccountEdit/5

[HttpPost]

public ActionResult AccountEdit(long? id, int iMonth, int iYear)

{

vw\_BankAccount bankAccountItem = null;

vw\_Account item = null;

vw\_hm\_DecisionToObjectLink decisionLink = null;

try {

bool addBankAccount = !string.IsNullOrWhiteSpace(Request.Form["BankAccount.Number"]);

if (id.HasValue)

item = AccountRepository.Get(id.Value);

else

item = new vw\_Account();

item.IncomeMonthStart = new DateTime(iYear, iMonth, 1);

var itemTypeBefore = item.TypeCode;

var oldBankAccountId = item.BankAccountId;

TryUpdateModel(item, "Item", new string[] { "DateStart", "DateEnd", "ObjectId", "Responsible", "TypeId", "BankAccountId", "OrganizationFullName", "DocumentId" });

if (id.HasValue)

TryUpdateModel(item, "Item", new string[] { "StatusId" });

decisionLink = oldDb.vw\_hm\_DecisionToObjectLinks.FirstOrDefault(m => m.Id == (item.DecisionLinkId ?? 0)) ?? new vw\_hm\_DecisionToObjectLink {

ObjectId = item.ObjectId.Value,

Updated = DateTime.Today,

Created = DateTime.Today

};

UpdateModel(decisionLink, "DecisionLink", new string[] { "Rate", "RateDateStart" });

if (addBankAccount)

{

bankAccountItem = BankAccountRepository.GetByNumber( Request.Form["BankAccount.Number"]);

if (bankAccountItem == null)

{

bankAccountItem = new vw\_BankAccount();

bankAccountItem.DateStart = item.DateStart;

bankAccountItem.TypeId = oldDb.FacetItems\_get ("fin\_BankAccount\_type").FirstOrDefault(m => m.Code == "payment").Id;

bankAccountItem.StatusId = oldDb.FacetItems\_get("fin\_Account\_status").FirstOrDefault(m => m.Code == "created").Id;

UpdateModel(bankAccountItem, "BankAccount", new string[] { "OrganizationBankBIK", "OrganizationBankInn",

"OrganizationBankKpp", "OrganizationBankOgrn", "OwnerOrganizationId", "OrganizationFullName",

"DateStart", "DateEnd", "Number", "OrganizationBankFullName", "OrganizationBankShortName" });

}

item.BankAccountId = bankAccountItem.Id;

}

else

{

bankAccountItem = BankAccountRepository.Get(item.BankAccountId);

}

if (addBankAccount)

{

BankAccountRepository.Edit(bankAccountItem, false);

}

AccountRepository.Edit(item,false,false);

AccountRepository.SaveStatus(item);

var typeCode = oldDb.FacetItems\_get("fin\_Account\_type").FirstOrDefault(d => d.Id == item.TypeId).Code;

if (Helpers.AppHelper.User.IsInRole("region.bigRepairOperator") && (!string.IsNullOrWhiteSpace(itemTypeBefore) && itemTypeBefore == "individual" || typeCode == "individual"))

throw new AIS.HM.Model.HMNoRightsException("Вам не доступно создание с типом специальный счёт в банке.");

else if (Helpers.AppHelper.User.IsInRole("region.housingInspection") && (!string.IsNullOrWhiteSpace(itemTypeBefore) && itemTypeBefore != "individual" || typeCode != "individual"))

throw new AIS.HM.Model.HMNoRightsException("Вамдоступносозданиесчетовтолькостипомспециальныйсчётвбанке.");

if (id.HasValue && item.BankAccountId != oldBankAccountId) {

var listTakenOperations = from o in oldDb.vw\_Operations

join g in oldDb.vw\_GroupOperations on o.GroupOperationId equals g.Id

where g.StatusCode == "goTaken" && o.AccountId == item.Id

select o;

if (listTakenOperations.Any())

throw new AIS.HM.Model.HMException("Нельзя изменить банковский счет у счета МКД, так как на нем есть учтенные групповые операции.");

}

string strDocId = Request.Form["Item.DocumentId"];

if (!string.IsNullOrEmpty(strDocId))

{

int documentId = int.Parse(strDocId);

var decision = oldDb.vw\_hm\_Decisions.FirstOrDefault(m => m.DocumentId == documentId);

if (decision == null)

{

throw new AIS.HM.Model.HMException("Ненайденорешениеповыбранномудокументу.", "Item.DocumentId");

}

if (item.DecisionLinkId.HasValue)

{

if (decision.DocumentId != item.DocumentId)

{

decisionLink.DecisionId = decision.Id;

}

oldDb.SubmitChanges();

}

else

{

decisionLink.DecisionId = decision.Id;

oldDb.vw\_hm\_DecisionToObjectLinks.InsertOnSubmit(decisionLink);

oldDb.SubmitChanges();

item.DecisionLinkId = decisionLink.Id;

}

}

else

{

item.DecisionLinkId = null;

}

if (addBankAccount)

{

BankAccountRepository.Edit(bankAccountItem);

BankAccountRepository.Submit();

item.BankAccountId = bankAccountItem.Id;

}

AccountRepository.Edit(item);

AccountRepository.Submit();

return this.RedirectToActionOrBackLink("BankAccountDetails", new { id = item.BankAccountId });

}

catch(Exception e){

this.ProcessException(e);

return View(new AccountViewModel(AccountRepository, item, bankAccountItem, null, decisionLink));

}

}

// Отображает страницу с детальной информацией по счёту

// GET: /BigRepair/Account/AccountDetails

public ActionResult AccountDetails(int id)

{

var item = AccountRepository.Get(id);

vw\_BankAccount bankItem = null;

vw\_cmn\_Document document = AccountRepository.getAccountDocument(item.DocumentId);

if (item == null)

ModelState.AddModelError(string.Empty, "Банковскийсчётненайден");

else

bankItem = BankAccountRepository.Get(item.BankAccountId);

return View(new AccountViewModel(AccountRepository, item, bankItem, document));

}

// Удаляетсчёт

// POST: /BigRepair/Account/AccountDelete/5

[HttpPost]

public ActionResult AccountDelete(long id, string returnUrl)

{

var item = AccountRepository.Get(id);

if (item == null)

throw new ArgumentException("Счётненайден.");

if (ModelState.IsValid)

{

try

{

if (Helpers.AppHelper.User.IsInRole("null.admin") || Helpers.AppHelper.User.IsInRole("region.bigRepairOperator") && item.TypeCode != "individual" || Helpers.AppHelper.User.IsInRole("region.housingInspection") && item.TypeCode == "individual")

{

AccountRepository.Delete(item);

AccountRepository.Submit();

}

else

{

throw new AIS.HM.Model.HMNoRightsException();

}

}

catch (Exception ex)

{

ModelState.AddModelError(string.Empty, ex.Message);

this.StoreModelStateErrors();

}

}

else

{

this.StoreModelStateErrors();

}

return this.RedirectToActionOrBackLink(returnUrl ?? Url.Action("BankAccountDetails", new { id = item.BankAccountId }));

}

// Отображает страницу редактирования банковского счёта

// GET: /BigRepair/Account/BankAccountEdit

public ActionResult BankAccountEdit(int? id)

{

vw\_BankAccount item;

if (id.HasValue)

item = BankAccountRepository.Get(id.Value);

else

item = new vw\_BankAccount() { DateStart = DateTime.Now };

if (item == null)

throw new ArgumentException("Счётненайден.");

return View(new BankAccountViewModel(BankAccountRepository).FillItem(item));

}

// Редактируетбанковскийсчёт

// POST: /BigRepair/Account/BankAccountEdit

[HttpPost]

public ActionResult BankAccountEdit(int? id, string foo)

{

vw\_BankAccount item;

if (id.HasValue)

item = BankAccountRepository.Get(id.Value);

else

item = new vw\_BankAccount();

TryUpdateModel(item, "Item", new string[] { "TypeId", "DateStart", "DateEnd", "OwnerOrganizationId", "Number", "OrganizationFullName",

"OrganizationBankFullName", "OrganizationBankShortName", "OrganizationBankBIK", "OrganizationBankInn", "OrganizationBankKpp", "OrganizationBankOgrn" });

if (id.HasValue)

TryUpdateModel(item, "Item", new string[] { "StatusId" });

if (ModelState.IsValid)

{

try

{

BankAccountRepository.SaveStatus(item);

}

catch (Exception ex)

{

ModelState.AddModelError(string.Empty, ex.Message);

this.StoreModelStateErrors();

}

try

{

if (User.IsInRole("region.bigRepairOperator") && oldDb.vw\_Account.Any(d => d.BankAccountId == item.Id && d.TypeCode == "individual"))

throw new AIS.HM.Model.HMNoRightsException("На данном банковском счёте есть счета МКД с типом специальный счёт в банке");

else if (User.IsInRole("region.housingInspection") && oldDb.vw\_Account.Any(d => d.BankAccountId == item.Id && d.TypeCode != "individual"))

throw new AIS.HM.Model.HMNoRightsException("Выможетеработатьсоспециальнымисчетамивбанке. На данном банковском счёте есть счета иного типа");

BankAccountRepository.Edit(item);

BankAccountRepository.Submit();

}

catch (Exception ex)

{

ModelState.AddModelError(string.Empty, ex.Message);

return View(new BankAccountViewModel(BankAccountRepository).FillItem(item));

}

return this.RedirectToActionOrBackLink("BankAccountDetails", new { id = item.Id });

}

return View(new BankAccountViewModel(BankAccountRepository).FillItem(item));

}

// Отображает страницу с реестром банковских счетов

// GET: /BigRepair/Account/BankAccountList

public ActionResult BankAccountList(BankAccountFilter filter)

{

return View(new BankAccountViewModel(BankAccountRepository, filter).FillList());

}

// Отображает страницу со списком счетов МКД

// GET: /BigRepair/Account/AccountList

public ActionResult AccountList(AccountFilter filter)

{

return View(new AccountViewModel(AccountRepository, filter, oldDb).FillList());

}

// Отображает страницу со списком счетов МКД

// GET: /BigRepair/Account/AccountSuspiciousList

public ActionResult AccountSuspiciousList(AccountFilter filter)

{

var list = oldDb.vw\_SuspiciousAccount;

var listAccount = ((filter != null) && (filter.flt != null)) ? list.Where(m => SqlMethods.Like(m.ObjectName, filter.flt.PrepareForLike()) || SqlMethods.Like(m.BankAccountNumber, filter.flt.PrepareForLike())) : list;

if (filter != null && !String.IsNullOrEmpty(filter.problemCode))

listAccount = listAccount.Where(m => m.ProblemCode == filter.problemCode);

listAccount = listAccount.OrderBy(m => m.ProblemSort).ThenBy(m => m.ProblemCode);

var optionsList = from f in oldDb.FacetItems\_get("fin\_SuspiciousAccount")

select new {Value = f.Code, Text = f.Name+" ("+ list.Count(m => m.ProblemCode == f.Code) +")"};

return View(new AccountViewModel(AccountRepository, filter, oldDb) {

SuspiciousList = new PaginatedList<vw\_SuspiciousAccount>(listAccount, (filter != null) ? filter.page : null),

AccountProblemsOptions = new SelectList(optionsList, "Value", "Text", filter.problemCode)

});

}

// Отображает страницу со списком счетов МКД

// GET: /BigRepair/Account/BankAccountSuspiciousList

public ActionResult BankAccountSuspiciousList(BankAccountFilter filter)

{

var list = oldDb.vw\_SuspiciousBankAccount;

var listBankAccount = ((filter != null) && (filter.flt != null)) ? list.Where(m => SqlMethods.Like(m.Number, filter.flt.PrepareForLike()) || SqlMethods.Like(m.OrganizationBankShortName, filter.flt.PrepareForLike()) || SqlMethods.Like(m.OrganizationBankFullName, filter.flt.PrepareForLike())) : list;

if (filter != null && !String.IsNullOrEmpty(filter.problemCode))

listBankAccount = listBankAccount.Where(m => m.ProblemCode == filter.problemCode);

listBankAccount = listBankAccount.OrderBy(m => m.ProblemSort).ThenBy(m => m.ProblemCode);

var optionsList = from f in oldDb.FacetItems\_get("fin\_SuspiciousBankAccount")

select new { Value = f.Code, Text = f.Name + " (" + list.Count(m => m.ProblemCode == f.Code) + ")" };

return View(new BankAccountViewModel(BankAccountRepository, filter) {

SuspiciousList = new PaginatedList<vw\_SuspiciousBankAccount>(listBankAccount, (filter != null) ? filter.page : null),

BankAccountProblemsOptions = new SelectList(optionsList, "Value", "Text", filter.problemCode)

});

}

// Отображает страницу с детальной информацией по счёту

// GET: /BigRepair/Account/BankAccountDetails

public ActionResult BankAccountDetails(int id, BankAccountFilter filter)

{

var item = BankAccountRepository.Get(id);

if (item == null)

ModelState.AddModelError(string.Empty, "Банковскийсчётненайден");

return View(new BankAccountViewModel(BankAccountRepository, filter).FillItem(BankAccountRepository.Get(id)).FillListAccounts(AccountRepository));

}

// Удаляетбанковскийсчёт

// POST: /BigRepair/Account/BankAccountDelete/5

[HttpPost]

public ActionResult BankAccountDelete(int id, string returnUrl)

{

var item = BankAccountRepository.Get(id);

if (item == null)

ModelState.AddModelError(string.Empty, "Счётненайден.");

if (ModelState.IsValid)

{

try

{

if (User.IsInRole("region.bigRepairOperator") && oldDb.vw\_Account.Any(d => d.BankAccountId == item.Id && item.TypeCode == "individual"))

throw new AIS.HM.Model.HMNoRightsException("На данном банковском счёте есть счета МКД с типом специальный счёт в банке");

if (User.IsInRole("region.housingInspection") && oldDb.vw\_Account.Any(d => d.BankAccountId == item.Id && item.TypeCode != "individual"))

throw new AIS.HM.Model.HMNoRightsException("Выможетеработатьсоспециальнымисчетамивбанке. На данном банковском счёте есть счета иного типа");

BankAccountRepository.Delete(item);

BankAccountRepository.Submit();

}

catch (Exception ex)

{

ModelState.AddModelError(string.Empty, ex.Message);

this.StoreModelStateErrors();

}

}

else

{

this.StoreModelStateErrors();

}

return this.RedirectToActionOrBackLink(returnUrl ?? Url.Action("BankAccountList"));

}

public ActionResult AccountControl(AccountFilter filter)

{

return View(new AccountViewModel(AccountRepository, filter, oldDb).FillObjectList());

}

[HttpPost]

public ActionResult ObjectList(string term)

{

var list = ObjectRepository.GetListByUseForBigRepair(term);

return Json(list.Select(d => new { id = d.Id, label = d.StructureText }));

}

[HttpPost]

public ActionResult OrganizationList(string term, string type)

{

int? localTypeId = null;

var list = OrganizationRepository.GetList(

typeId: localTypeId,

active: null,

filled: true,

filter: term

);

return Json(list.Select(d => new { id = d.Id, label = d.ShortName + " (" + d.Inn + ")" }));

}

[HttpPost]

public ActionResult DocumentList(int id, int aId, DateTime date, AccountFilter filter, int methodId)

{

var bankAccount = BankAccountRepository.Get(id);

var account = AccountRepository.Get(aId);

var docId = account != null && account.DocumentId.HasValue ? account.DocumentId.Value : -1;

var rep = new WinterPreparations.Areas.Personal.Models.DocumentRepository();

int? orgId = filter.identityOrgId;

string fundMethodCode = oldDb.FacetItems\_get("fin\_Account\_type").FirstOrDefault(m => m.Id == methodId).Code;

// decree

var list = from d in oldDb.vw\_cmn\_Documents

join deci in oldDb.vw\_hm\_Decisions on d.Id equals deci.DocumentId

where (d.OrgId == null || d.OrgId == orgId)

&& d.TypeId == rep.GetType("LocalGovernmentDecree").Id

&& d.Date <= date

&& deci.FundMethodCode == fundMethodCode

|| d.Id == docId

&& deci.FundMethodCode == fundMethodCode

select d

;

// ownerMeeting

if (account != null)

list = list.Union(

from d in oldDb.vw\_cmn\_Documents

join m in oldDb.vw\_hm\_Decisions on d.Id equals m.DocumentId

join n in oldDb.vw\_hm\_DecisionToObjectLinks on m.Id equals n.DecisionId

where (account == null || n.ObjectId == account.ObjectId)

&& d.TypeId == rep.GetType("ownerMeeting").Id

&& d.Date <= date.AddMonths(1)

&& m.FundMethodCode == fundMethodCode

|| d.Id == docId

&& m.FundMethodCode == fundMethodCode

select d

);

return Json(Option.PrepareList(list));

}

}

}

Программныйкодмодуля ObjectDetails

Файл PassportController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using WinterPreparations.Models;

using WinterPreparations.Areas.ObjectDetails.Models;

using System.Data.Linq.SqlClient;

namespace WinterPreparations.Areas.ObjectDetails.Controllers

{

[Authorize]

public class PassportController : Controller

{

[NonAction]

bool CanEdit(bool isAdmin, int orgId, vw\_cmn\_Object objectItem)

{

if (objectItem == null) return false;

if (Helpers.AppHelper.User.IsInRole("supplier.") && oldDb.CanManageObject(orgId, objectItem.Id, DateTime.Now, new string[] {

"resourceService",

"manage",

"properManagement"

}))

return true;

else if (Helpers.AppHelper.User.IsInRole("localgov.") &&

oldDb.CanManageObject(orgId, objectItem.Id, DateTime.Now, new string[] {

"manage",

"properManagement"

}))

return true;

else if (Helpers.AppHelper.User.IsInRole("org.") &&

oldDb.CanManageObject(orgId, objectItem.Id, DateTime.Now, new string[] {

"manage",

"properManagement",

"contract"

}))

return true;

else if (Helpers.AppHelper.User.IsInRole("null."))

return true;

else return false;

}

[NonAction]

public bool IsEditable(int id)

{

vw\_Parameter parentItem = new vw\_Parameter();

vw\_Parameter item = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == id);

if (item.ParentId.HasValue)

parentItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == item.ParentId);

if(parentItem != null && parentItem.HasListChildren)

return false;

if (!item.IsData)

return false;

return true;

}

[HttpPost]

public ActionResult SaveArray(int objectId, int orgId, bool isAdmin, DateTime dateValue, List<string> pId)

{

int currentPId = 0;

int? currentSortX = null;

int? currentSortY = null;

string errorMsg = "";

int integer;

DateTime updated = DateTime.Now;

Dictionary<string, string []> changedValues = new Dictionary<string, string []>();

if (pId == null) pId = new List<string>();

try

{

foreach (string i in pId)

{

currentSortX = null;

currentSortY = null;

string[] s = i.Split(new char[] { '\_' });

currentPId = int.Parse(s[0]);

if (int.TryParse(s[1], out integer))

currentSortX = int.Parse(s[1]);

if (int.TryParse(s[2], out integer))

currentSortY = int.Parse(s[2]);

vw\_ParameterValue temp = new vw\_ParameterValue() { ParameterId = currentPId, ObjectId = objectId, DateStart = dateValue, SortX = currentSortX, SortY = currentSortY };

UpdateModel(temp, "p\_" + i, new string[] {

"Value"

});

vw\_ParameterValue item = oldDb.vw\_ParameterValues.FirstOrDefault(d =>

d.ParameterId == currentPId &&

d.ObjectId == objectId &&

(currentSortX.HasValue ? d.SortX == currentSortX : d.SortX == null) &&

(currentSortY.HasValue ? d.SortY == currentSortY : d.SortY == null) &&

d.DateStart == dateValue

);

vw\_Parameter parameter = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == currentPId);

bool isError = false;

string pValue = string.Empty;

ParseValue(temp.Value, out pValue, parameter.DataType, out isError);

string formatRus = string.Empty;

string errorString = null;

if (temp.Value != pValue || item == null || item.Value != temp.Value) {

if (isError)

{

switch (parameter.DataType)

{

case "int": formatRus = "\"целоечисло\""; break;

case "decimal": formatRus = "\"дробноечисло\""; break;

case "date": formatRus = "\"дата\""; break;

case "year": formatRus = "\"год\""; break;

default: formatRus = "\"строка\""; break;

}

errorString = "несоотв. типу " + formatRus;

}

if (!string.IsNullOrEmpty(errorString))

changedValues.Add("p\_" + i + ".Value", new string[] { pValue, errorString });

temp.Value = pValue;

}

if (item == null)

{

if (!string.IsNullOrEmpty(temp.Value))

{

item = new vw\_ParameterValue()

{

ObjectId = objectId,

ParameterId = currentPId,

SortX = currentSortX,

SortY = currentSortY,

DateStart = dateValue,

Value = temp.Value,

Updated = updated,

Priority = 1

};

changedValues.Add("p\_" + i + ".Value", new string[] { pValue, errorString });

if (oldDb.vw\_ParameterValues.GetOriginalEntityState(item) == null)

oldDb.vw\_ParameterValues.InsertOnSubmit(item);

}

}

else

if (!(string.IsNullOrEmpty(item.Value) && string.IsNullOrEmpty(temp.Value)) || !(item.Value == temp.Value))

{

item.Value = temp.Value;

item.Updated = updated;

item.Priority = 1;

changedValues.Add("p\_" + i + ".Value", new string[] { pValue, errorString });

}

}

if (!CanEdit(isAdmin, orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == objectId)))

throw new AIS.HM.Model.HMNoRightsException();

oldDb.SubmitChanges();

oldDb.FillPart\_RecalcByObjectIdAndMonth(objectId, dateValue);

}

catch (AIS.HM.Model.HMException e)

{

vw\_Parameter parameter = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == currentPId);

if (currentSortY.HasValue)

parameter.Number = parameter.Number.Replace("{y}", currentSortY.Value.ToString());

if (currentSortX.HasValue)

parameter.Number = parameter.Number.Replace("{x}", currentSortX.Value.ToString());

errorMsg = "Ошибкивразделе \"" + parameter.GroupName.Split(new string[] { "." }, StringSplitOptions.None)[0] + "\", пункт " + parameter.Number;

if (Request.IsLocal || e is AIS.HM.Model.HMException)

errorMsg += ": " + e.Message;

return JsonResponse(new AIS.HM.Model.HMException(errorMsg));

}

return JsonResponse(changedValues);

}

public ActionResult Details(int id, PassportFilterModel filter)

{

//if has no data in current mount changed date filter

if (!filter.hasDate) filter.dateValue = new DateTime(filter.dateValue.Year, filter.dateValue.Month - 1, filter.dateValue.Day);

vw\_cmn\_Object objectItem = oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == id);

filter.id = id;

var item = oldDb.vw\_1468Filling\_getByDate(filter.dateValue).FirstOrDefault(a => a.ObjectId == id);

ViewBag.FillPercentage = item != null ? item.FillPercentage : null;

ViewBag.UpdateDate = item != null ? (DateTime?)item.Updated : null;

ViewBag.canEdit = CanEdit(filter.isAdmin, filter.orgId, objectItem);

ViewBag.filter = filter;

ViewBag.address = objectItem.StructureText;

ViewBag.list = oldDb.vw\_ParameterGroups.Where(d => d.ObjectTypeId == objectItem.TypeId).OrderBy(d => d.Sort).ToList();

bool display1468OwnOnly = Helpers.ConfigurationHelper.GetDefault("common.Display1468OwnOnly", false) ?? false;

bool can = true;

if (display1468OwnOnly)

{

if (filter.isLocalGov)

{

can = oldDb.IsObjectInLocalGov(Helpers.AppHelper.Identity.OrgId, filter.id);

}

else if (!(filter.isLocalGov || filter.isAdmin || filter.isRegion))

{

can = oldDb.CanManageObject(Helpers.AppHelper.Identity.OrgId, filter.id, DateTime.Now);

}

}

if (!can)

{

this.ProcessException(new AIS.HM.Model.HMNoRightsException(filter.isLocalGov ? "Объект не находится на территории Вашего МО." : "Вы не управляете данным объектом."));

this.StoreModelStateErrors();

return this.RedirectToActionOrBackLink(Url.Action("List", "Object", new { area = "Personal" }));

}

return View();

}

public ActionResult GetRows(int objectId, int? parentId, int? groupId, int? sortX, int? sortY, PassportFilterModel filter)

{

ViewBag.filter = filter;

var list = from d in oldDb.vw\_Parameters

where (parentId.HasValue ? d.ParentId == parentId : d.ParentId == null) &&

(groupId.HasValue ? d.GroupId == groupId : d.GroupId == null)

orderby d.Sort, d.ListSort, d.ListSortY

select d;

var valList = (from d in oldDb.vw\_ParameterValues

join l in list on d.ParameterId equals l.Id

where d.ObjectId == objectId &&

d.DateStart == filter.dateValue

select d).ToList();

for (int i = 0; i < valList.Count; i++)

{

var item = valList[i];

var parameters = list.SingleOrDefault(el => el.Id == item.ParameterId);

}

if (sortX.HasValue)

valList = valList.Where(d => d.SortX == sortX).ToList();

if (sortY.HasValue)

valList = valList.Where(d => d.SortY == sortY).ToList();

valList = valList.OrderBy(d => d.SortX).ThenBy(d => d.SortY).ToList();

ViewBag.canEdit = CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == objectId));

ViewBag.list = list.ToList();

ViewBag.valList = valList.ToList();

ViewBag.objectId = objectId;

return View();

}

public ActionResult Edit(PassportFilterModel filter, int parameterId, int? sort, int? sortX, int? sortY)

{

vw\_Parameter parameterItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == parameterId);

vw\_ParameterValue valueItem;

try

{

//if has list child get child item

if (parameterItem.HasListChildren == true)

parameterItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.ParentId == parameterId);

valueItem = oldDb.vw\_ParameterValues.FirstOrDefault(d =>

d.ParameterId == parameterItem.Id &&

d.ObjectId == filter.id &&

(sortX.HasValue ? d.SortX == sortX : d.SortX == null) &&(sortY.HasValue ? d.SortY == sortY : d.SortY == null) &&d.DateStart == filter.dateValue

);

ViewBag.IsEdit = true;

if (valueItem == null)

{

valueItem = new vw\_ParameterValue()

{

ObjectId = filter.id,

ParameterId = parameterItem.Id,

DateStart = filter.dateValue,

SortX = sortX,

SortY = sortY

};

ViewBag.IsEdit = false;

}

ViewBag.filter = filter;

ViewBag.valueItem = valueItem;

ViewBag.parameterItem = parameterItem;

return View();

}

catch (Exception ex)

{

this.ProcessException(ex);

}

return View();

}

[HttpPost]

public ActionResult Edit(PassportFilterModel filter, int? parameterId, string foo, int? sortX, int? sortY)

{

vw\_Parameter parameterItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == parameterId);

vw\_ParameterValue valueItem;

try

{

if (parameterItem.HasListChildren == true)

parameterItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.ParentId == parameterId);

valueItem = oldDb.vw\_ParameterValues.FirstOrDefault(d =>

d.ParameterId == parameterItem.Id && d.ObjectId == filter.id &&

(sortX.HasValue ? d.SortX == sortX : d.SortX == null) &&

(sortY.HasValue ? d.SortY == sortY : d.SortY == null) &&

d.DateStart == filter.dateValue);

if (valueItem == null)

valueItem = new vw\_ParameterValue()

{

ObjectId = filter.id,

DateStart = filter.dateValue,

ParameterId = parameterItem.Id,

Updated = DateTime.Now

};

TryUpdateModel(valueItem, "valueItem", new[] {

"Value",

"Description"

});

string[] nums = valueItem.Value.Split(new char[] { ',', ';', '.' });

string description = valueItem.Description;

Dictionary<string, string> errors = new Dictionary<string, string>();

// parse

List<int> intNums = new List<int>();

foreach (var n in nums)

{

string s = n.Trim();

int dashPos = s.IndexOf("-");

// interval

if (dashPos >= 0)

{

int start = 0, end = 0;

int.TryParse(s.Substring(0, dashPos), out start);

int.TryParse(s.Substring(dashPos + 1, s.Length - dashPos - 1), out end);

if (end < start)

{

dashPos = start;

start = end;

end = dashPos;

}

if (start > 0 && end > 0)

for (var i = start; i <= end; i++)

{

if (!intNums.Contains(i))

intNums.Add(i);

}

else if (start <= 0 || end <= 0)

errors.Add(s, "Граница интервала должна быть положительной");

else if (end < start)

errors.Add(s, "Граница интервала должна быть положительной");

}

// just number

else

{

int num = 0;

if (int.TryParse(s, out num))

{

if (!intNums.Contains(num))

intNums.Add(num);

}

else

errors.Add(s, "Ошибка значения: не удалось преобразовать номер к целому числу");

}

}

var result = oldDb.vw\_ParameterValues.Where(d => d.ParameterId == parameterItem.Id && d.ObjectId == filter.id);

foreach (int i in intNums)

{

valueItem = oldDb.vw\_ParameterValues.FirstOrDefault(d =>

d.ParameterId == parameterItem.Id &&d.ObjectId == filter.id &&

(sortX.HasValue ? d.SortX == sortX : d.SortX == null) &&

(sortY.HasValue ? d.SortY == sortY : d.SortY == null) &&

d.DateStart == filter.dateValue

);

string newDescr = "";

//create new

if (!string.IsNullOrEmpty(description))

if (description.Trim().Contains("№"))

newDescr = description + " " + i.ToString();

else

newDescr = description;

else

newDescr = null;

if(valueItem == null)

valueItem = new vw\_ParameterValue()

{

ObjectId = filter.id,

DateStart = filter.dateValue,

ParameterId = parameterItem.Id,

Description = newDescr,

Updated = DateTime.Now

};

//update current

valueItem.Value = i.ToString();

valueItem.Description = newDescr;

valueItem.Priority = 1;

if (!ModelState.IsValid)

throw new AIS.HM.Model.HMException();

if (!CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == valueItem.ObjectId)))

throw new AIS.HM.Model.HMNoRightsException();

if (oldDb.vw\_ParameterValues.GetOriginalEntityState(valueItem) == null)

{

if (!sortX.HasValue)

valueItem.SortX = Convert.ToInt32(valueItem.Value);

else

{

valueItem.SortX = sortX;

valueItem.SortY = Convert.ToInt32(valueItem.Value);

}

oldDb.vw\_ParameterValues.InsertOnSubmit(valueItem);

}

else

{

var parentParameterItem = oldDb.vw\_Parameters.FirstOrDefault(d =>d.Id == parameterItem.ParentId);

if (parentParameterItem != null)

if (sortY.HasValue)

{

valueItem.SortX = sortX;

valueItem.SortY = Convert.ToInt32(valueItem.Value);

oldDb.RecalculateChildrenSort(valueItem.Id, null, valueItem.Value);

}

else

{ oldDb.RecalculateChildrenSort(valueItem.Id, valueItem.Value, null);

valueItem.SortX = Convert.ToInt32(valueItem.Value);

}

}

}

if(oldDb.vw\_ParameterValues.GetModifiedMembers(valueItem).Any())

valueItem.Updated = DateTime.Now;

oldDb.SubmitChanges();

oldDb.FillPart\_RecalcByObjectIdAndMonth(filter.id, filter.dateValue);

ViewBag.valueItem = valueItem;

ViewBag.paramterItem = parameterItem;

if (Request.IsAjaxRequest())

{

return Json(false);

}

return RedirectToAction("Details", filter.ToRouteValueDictionaryAsBase());

}

catch (Exception ex)

{

this.ProcessException(ex);

}

return View();

}

public ActionResult Delete(PassportFilterModel filter, int objectId, int parameterId, int? sortX, int? sortY, DateTime dateStart)

{

vw\_Parameter parameterItem = oldDb.vw\_Parameters.FirstOrDefault(d => d.Id == parameterId);

vw\_ParameterValue valItem = oldDb.vw\_ParameterValues.FirstOrDefault(d =>

d.ParameterId == parameterItem.Id &&d.ObjectId == objectId &&

(sortX.HasValue ? d.SortX == sortX : d.SortX == null) &&

(sortY.HasValue ? d.SortY == sortY : d.SortY == null) &&

d.DateStart == dateStart);

if (valItem != null)

{

if (!CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == objectId)))

throw new AIS.HM.Model.HMNoRightsException();

oldDb.vw\_ParameterValues.DeleteOnSubmit(valItem);

oldDb.SubmitChanges();

}

foreach (var p in oldDb.vw\_Parameters.Where(d => d.ParentId == parameterItem.Id))

{

Delete(filter, objectId, p.Id, sortX, sortY, dateStart);

}

oldDb.FillPart\_RecalcByObjectIdAndMonth(objectId, dateStart);

if (Request.IsAjaxRequest())

return Json(false);

return null;

}

[HttpPost]

public ActionResult DeleteDateData(int id, PassportFilterModel filter)

{

try

{

if (!CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == id)))

throw new AIS.HM.Model.HMNoRightsException();

if (oldDb.vw\_ParameterValues.Where(d => d.ObjectId == id && d.DateStart == filter.dateValue).Any())

{

db.Connection.Open();

db.SetUserId(Helpers.AppHelper.Identity.Id);

db.ParameterValues\_delete(id, filter.dateValue);

throw new AIS.HM.Model.HMException("Данныебылиуспешноудалены", false);

}

else

throw new AIS.HM.Model.HMException("По данному месяцу данные не заносились");

}

catch (Exception ex)

{

this.ProcessException(ex);

this.StoreModelStateErrors();

}

vw\_cmn\_Object objectItem = oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == id);

filter.id = id;

ViewBag.filter = filter;

ViewBag.address = objectItem.StructureText;

ViewBag.list = oldDb.vw\_ParameterGroups.Where(d => d.ObjectTypeId == objectItem.TypeId).OrderBy(d => d.Sort).ToList();

return RedirectToAction("Details", filter);

}

[HttpPost]

public ActionResult ImportObjectDetails(PassportFilterModel filter)

{

try

{

if (!CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == filter.id)))

throw new AIS.HM.Model.HMNoRightsException();

oldDb.Connection.Open();

oldDb.SetUserId(Helpers.AppHelper.Identity.Id);

oldDb.od\_Report\_ElPas\_import(filter.id, filter.dateValue);

throw new AIS.HM.Model.HMException("Данныеуспешноимпортированы", false);

}

catch (Exception ex)

{

this.ProcessException(ex);

this.StoreModelStateErrors();

}

return RedirectToAction("Details", filter);

}

[HttpPost]

public ActionResult CopyObjectDetails(PassportFilterModel filter)

{

try

{

if (!CanEdit(filter.isAdmin, filter.orgId, oldDb.vw\_cmn\_Objects.FirstOrDefault(d => d.Id == filter.id)))

throw new AIS.HM.Model.HMNoRightsException();

DateTime dateFrom = filter.dateValue.AddMonths(-1);

if (!oldDb.vw\_ParameterValues.Any(pv => pv.ObjectId == filter.id && pv.DateStart == dateFrom))

throw new AIS.HM.Model.HMException(String.Format("Данныена {0:d} ещёнезаполнены", dateFrom));

db.Connection.Open();

db.SetUserId(Helpers.AppHelper.Identity.Id);

db.ParameterValues\_copy(filter.id, dateFrom, filter.dateValue);

throw new AIS.HM.Model.HMException("Данныеуспешноимпортированы", false);

}

catch (Exception ex)

{

this.ProcessException(ex);

this.StoreModelStateErrors();

}

return RedirectToAction("Details", filter);

}

}

}