Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 – Программная инженерия
Направленность (профиль) образовательной программы: Компьютерное моделирование

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«15» 07 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Разработка программного обеспечения по осуществлению мониторинга исполнения программных мероприятий администрации города Благовещенска

Исполнитель
студент группы 857-ом _____ (подпись, дата) В.Н. Шульга

Руководитель
доцент, канд. техн. наук _____ 15.07.2020 (подпись, дата) А.В. Бушманов

Руководитель научного
содержания программы
магистратуры _____ (подпись, дата) И.Е. Ерёмин

Нормоконтроль _____ (подпись, дата) В.В. Ерёмина

Рецензент _____ (подпись, дата) И.С. Вирта

Благовещенск 2020

РЕФЕРАТ

Магистерская диссертация содержит 97 страниц, 20 рисунков, 6 приложений, 36 литературных источников.

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, МОДЕЛИРОВАНИЕ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ, ОТЧЁТЫ, ПРОЕКТИРОВАНИЕ, UML

Целью данной выпускной квалификационной работы является разработка программного обеспечения для оперативного мониторинга исполнения программных мероприятий администрации города Благовещенска с графическим интерфейсом пользователя и возможностью хранения информации в реляционной базе данных. Актуальность задачи заключается в том, что программное обеспечение жизненно необходимо управлению экономического развития и инвестиций, в то же время бюджет города не предполагает закупку подобного программного обеспечения.

В процессе диссертационной работы проведено обследование администрации и подведомственных учреждений на предмет выявления бизнес-процессов происходящих при взаимодействии подразделений. Подготовлено техническое задание на разработку ведомственного программного обеспечения в соответствии с которым реализовано программное обеспечение, осуществлен переход от абстракций и алгоритмов предметной области, предоставленных в техническом задании, к терминам объектно-ориентированной модели, разработана и реализована логическая структура базы данных, разработаны диаграммы IDEF0 и модели на языке UML, осуществлена их трансляция на язык PHP с непрерывной синхронизацией с реляционной базой данных. Разработан и апробирован графический интерфейс пользователя.

Разработанное программное обеспечение внедрено и прошло опытную эксплуатацию, по результату которой подготовлено техническое задание на развитие информационной системы.

Результаты исследования были апробированы на двух научных конференциях и опубликованы в трёх научных сборниках.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Президент Российской Федерации Владимир Владимирович Путин в посланиях Федеральному собранию РФ в 2013-2017 годах говорил о необходимости реформирования деятельности и усиления роли контрольных и надзорных ведомств. В частности, глава государства поручил создать систему контрольно-надзорной деятельности, основанной на новых принципах, позволяющих повысить ее результативность, одновременно снижая административное давление на общество. При этом необходимо обеспечить применение системы оценки рисков, позволяющей сосредоточить усилия ревизионного блока на вопросах, требующих особого внимания.

Главным результатом реализации поручений Президента стало утверждение Правительством Концепции повышения эффективности контрольно-надзорной деятельности органов государственной власти и органов местного самоуправления. Данная концепция предполагает комплексное и поэтапное изменение нормативно-правовой базы в сфере контроля и подготовку рамочного закона о федеральном, региональном и муниципальном контроле. Разработаны проект Федерального закона «Об основах государственного и муниципального контроля (надзора) в Российской Федерации» и Концепция развития систем внутреннего финансового контроля, внутреннего аудита и оценки качества финансового менеджмента в секторе государственного и муниципального управления.

Нормативная правовая база (предпосылки к разработке программного обеспечения)

1. Бюджетный кодекс Российской Федерации

2. Федеральный закон от 28.06.2014 № 172-ФЗ «О стратегическом планировании в Российской Федерации»

3. Распоряжение администрации города Благовещенска от 07.07.2015 № 117р «о разработке программного обеспечения для осуществления оперативно-

го мониторинга исполнения программных мероприятий администрации города Благовещенска».

Статья 179. Бюджетного кодекса Российской Федерации «Государственные программы Российской Федерации, государственные программы субъекта Российской Федерации, муниципальные программы» говорит нам, что муниципальные программы, объёмы ассигнований, порядок и сроки их реализации утверждаются муниципальным правовым актом. Ежегодно по каждой муниципальной программе проводится оценка эффективности её реализации. Порядок оценки утверждается муниципальным правовым актом.

Новации законодательства требуют изменения методов сбора и проведения анализа данных в контрольной деятельности, что, на наш взгляд, невозможно без своевременного внедрения соответствующих информационных систем. Для решения этой задачи было принято решение о разработке программного обеспечения «осуществления оперативного мониторинга исполнения программных мероприятий администрации города Благовещенска».

Цель диссертационной работы состоит в разработке программного обеспечения для оперативного мониторинга исполнения программных мероприятий администрации. Программное обеспечение позволит максимально точно представить информацию о текущей ситуации исполнения бюджета города Благовещенска и при необходимости принять управленческое решение.

В работе формулируются и решаются задачи, необходимые для достижения поставленной цели:

- анализ существующих технологий для разработки веб-приложения;
- язык разметки HTML, язык программирования PHP, стилизация CSS;
- логическое структурирование данных для информационной системы;
- разработка программной архитектуры;
- расширение функционала сетевой модели;
- внедрение системы в опытную эксплуатацию.

В магистерской диссертации использованы методы системного анализа и объектно-ориентированного подхода. Проектирование алгоритмического и программного обеспечения выполнено с использованием средств языка UML. Построены диаграммы классов, выполнена композиция и декомпозиция в нотации IDFE0. Разработана логическая структура базы данных, создана схема взаимодействия классов программного обеспечения. Проведено первичное заполнение справочников системы. При реализации использовалась среда web-разработки Geany, язык программирования PHP, язык стилей CSS.

Магистерская диссертация состоит из введения, трёх разделов, заключения, списка использованных источников, списка сокращений и приложений.

Во введении обосновывается актуальность темы, определяется цель работы и перечень решаемых задач.

В первом разделе рассматриваются основные принципы сетевого планирования. Дается обзор основных компонентов.

Во втором разделе производится сравнительный анализ нескольких сред для разработки веб-приложения. Далее проводится обзор шаблонов проектирования, выбор наиболее подходящего и обоснование.

В третьем разделе происходит выявление сущностей и построение диаграммы классов предметной области, а также описываются основные расчетные операции в виде диаграмм UML. На следующем этапе разрабатывается приложение в виде программного обеспечения.

В ходе разработки программного обеспечения были реализованы требования к информационной системе. Описаны основные механизмы для реализации системы. Представлено визуальное отображение текущей ситуации в виде отчётов, графиков.

1 АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К АВТОМАТИЗАЦИИ БИЗНЕС-ПРОЦЕССОВ

Этап анализа предметной области является общим при надлежащем проектировании и разработке любой информационной системы. Он позволяет собрать сведения об объекте и на их основе формировать видение будущего программного обеспечения, выбирать технологии создания, а так же определять набор предписаний к системе.

1.1 Трёхуровневая модель

Трёхуровневая модель (рисунок 1) – схема компонентов информационной системы, допускающая наличие в нем трёх элементов: клиента, модуль приложений (к которому подключен модуль клиентского приложения) и сервера баз данных (с которым соединяется модуль приложений).

Клиент (уровень клиента) – это интерфейсный (в настоящее время графический) компонент системы, предоставляемый конечному пользователю в виде визуальных (экранных) форм.

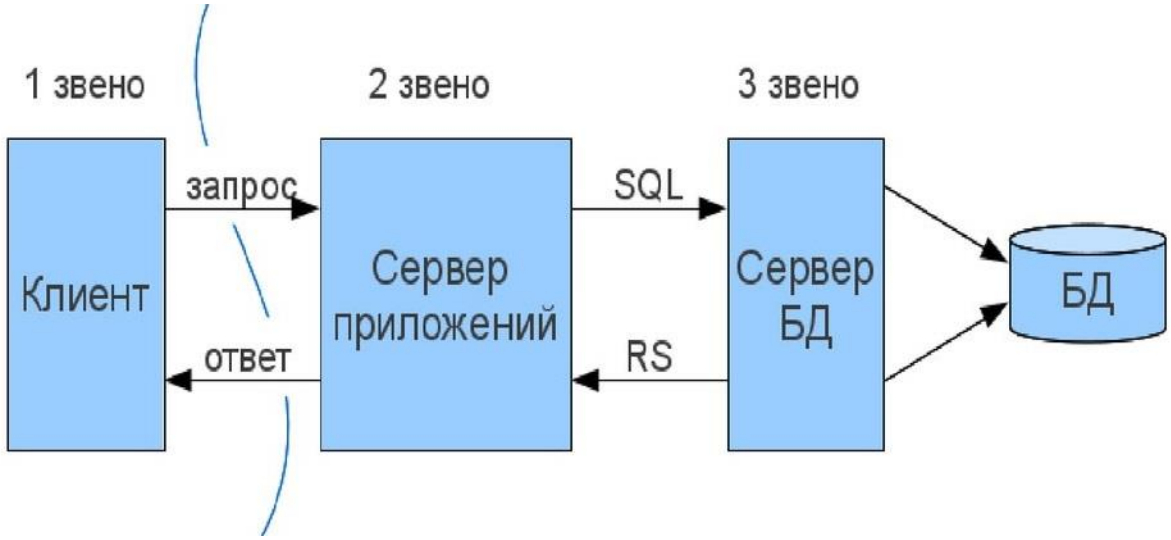


Рисунок 1 – Трёх уровневая модель.

Этот модуль не должен иметь никаких соединений с базой данных (по предписаниям безопасности и гибкости), быть высоконагруженным главной бизнес логикой (по предписаниям адаптивности) и хранить состояние модуля сервера приложений (по условиям надёжности). На этот уровень традиционно

выносится только простейшая бизнес-логика: процессор авторизации и аутентификации, проверка входных значений на правильность и соответствие формату, простые операции с информационной составляющей (сортировка, группировка, подсчёт итоговых показателей), уже отправленными на автоматизированное рабочее место (АРМ) клиента [16].

Сервер баз данных (третий уровень) осуществляет преимущественно обработку, хранение и передачу информацию в виде упорядоченных данных, который выносится на индивидуальный уровень, выполняются, зачастую, за счёт системы управления базами данных, соединение с модулем происходит только с уровня два, сервера приложений и то только посредством специализированных адаптеров.

Сервер приложений (связующий уровень) располагается за первым на втором ярусе, на нём содержится колоссальная часть бизнес-логики информационной системы. С наружи этого слоя содержатся только элементы, экспортируемые на автоматизированные рабочие места клиентов, а также элементы серверной логики, которые находятся в таблицах базы данных (триггеры, функции и хранимые процедуры). Работа данного модуля обеспечивается специальным программным обеспечением. Серверы приложений конструируются по такой схеме, чтобы присоединения к ним дополнительных модулей обеспечивало увеличение производительности информационной системы и не тянуло за собой серьёзные внесения изменений в исходный код используемого приложения.

В элементарных конфигурациях все модули информационной системы, или часть из них которые могут быть связаны в единый процессор в вычислительном кластере. В конфигурациях, выпущенных в производственную эксплуатацию, как правило, используется вычислительный узел монопольном режиме для серверов базы данных или кластер объединённых серверов баз данных, для действующих серверов уровня приложений – выделенная группа основных вычислительных мощностей, к которым напрямую подключаются клиенты.

По сравнению с файл-серверной схемой или клиент-серверной трёхуровневая архитектура даёт, как правило, наибольшую возможность к расширению

информационной системы. За счёт модернизации модулей сервера приложений и организации высокоскоростных соединений и передачу двух и более потоков информации в одном канале связи, большую гибкость. За счёт разобщенности уровней друг от друга, даёт широчайшие возможности применения инструкций по обеспечению безопасности и отказоустойчивости [17].

Наряду с этим, в сравнении с клиент-серверными информационными системами, использующими подключения по сокету или подключения непосредственно к серверу хранения базы данных, снижаются установленные требования к пропускной способности налаженных каналов связи между клиентским автоматизированным рабочим местом и серверной частью. Разработка систем, доступных из web-браузера по технологии тонкого клиента, зачастую, предполагает создание программного комплекса по трёхуровневой схеме. При этом, как правило, разработка приложений для трёхуровневых информационных систем сложнее, чем для классических клиент-серверных информационных систем, также присутствие драйвера связующего программного обеспечения повлечёт дополнительные затраты на администрирование таких информационных систем.

1.2 Анализ информационной структуры Муниципальной программы

Информационная структура стандартной Муниципальной программы представляет собой сложную схему, в которую входят: программные мероприятия, подпрограммы, основные мероприятия уровня программы и подпрограммы, цели, индикаторы, показатели, финансирование по каждому мероприятию. Ниже будет подробно описан каждый элемент Муниципальной программы.

1.2.1 Общие сведения о мониторинге программных мероприятий

Муниципальная программа – пакет документов социально-стратегического планирования, включающий комплекс запланированных к реализации мероприятий, объединённым по поставленным задачам, контрольным срокам исполнения, ответственными исполнителям, ресурсным обеспечением и обеспечивающих достижение поставленных ранее целей в установленные сро-

ки и решение определённых задач направленных в первую очередь на социально-экономическое развитие и в последствии развитие инвестиционной привлекательности муниципального образования города Благовещенска.

Исполнение муниципального бюджета в настоящее время во многом зависит от своевременного согласования конкретных действий всех заинтересованных структурных подразделений. Для обеспечения оперативного внесения изменений в программные мероприятия структурные подразделения должны оперативно получать достоверную информацию об доведении и объемах денежных средств до исполнителя. В то же время финансовый орган должен получать информацию от непосредственных исполнителей о кассовом исполнении доведенных средств. Зачастую бывает так, что средства доведены не в полном объёме, то в таком случае возникает задолженность или подрядчик не укладывается в сроки. В этих случаях необходимо принимать управленческое решение о переносе средств с одного мероприятия на другое.

На рисунке 2 изображена примерная схема движения информации (бизнес процесс) при работе с проектируемой системой.

Информационная система мониторинга исполнения программных мероприятий позволяет организовать слаженную работу всех ответственных исполнителей для обеспечения своевременного исполнения городского бюджета за счёт выявления проблемных точек и своевременного принятия решения о переопределении бюджетных средств. Использование информационной системы обеспечивает сто процентное исполнение бюджета в указанные сроки по каждому программному мероприятию, представление отчётов о плановых, текущих и завершенных мероприятиях.

Рисунок 2 – Схема процессов, протекающих при подготовке аналитических отчётов.

1.2.2 Анализ протекающих бизнес-процессов

Муниципальные программы социально-стратегического направления представляют собой очень сложную многократно ветвящуюся структуру с множеством возможных вариантов подчиненности. На рисунке 3 изображена упрощенная схема Муниципальной программы. Абсолютно каждая программа обязательно имеет в своём составе подпрограмму, координатора действий, ответственного исполнителя, цель и, хотя бы одно мероприятие, в свою очередь каждое мероприятие имеет финансирование по статьям бюджета и собственно ответственного исполнителя.

Муниципальная Подпрограмма выше стоящей муниципальной программы – комплекс объединённых по намеченным целям, установленным срокам исполнения и планируемым ресурсам программных мероприятий, направленных исключительно на решение вполне конкретных задач муниципального заказа. В то же время у одного мероприятия, как правило, ещё есть несколько исполнителей у каждого из них могут быть разные статьи поступления денежных средств.

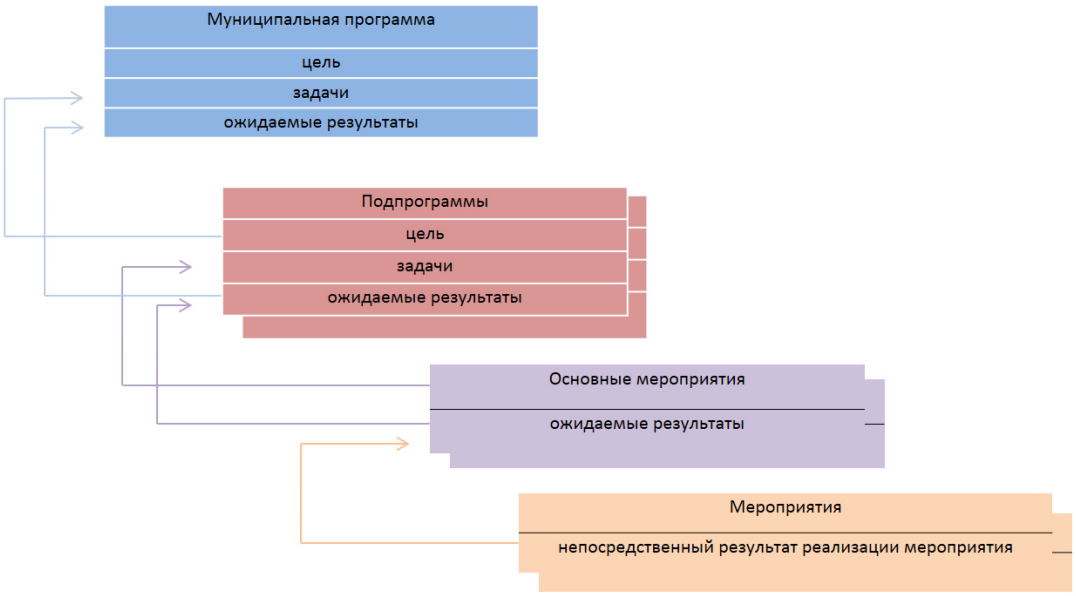


Рисунок 3 – Схема муниципальной программы.

Координатор Программы – определённое мэром уполномоченное лицо из числа заместителей мэра города Благовещенска; В роли муниципального заказчика программного мероприятия может выступать мэрия или структурное подразделение муниципального образования города Благовещенска, совершающее в пределах своих полномочий или компетенций разработку закрепленного за ним муниципальной задания и непосредственно ее реализацию, а также постоянную координацию деятельности в части исполнения муниципальной программы. В роли ответственного исполнителя выступает структурное подразделение, курирующее соответствующее направление профессиональной деятельности. Муниципальная программа имеет перед собой определённую цель, это может быть планируемый на определенный период реализации программного мероприятия конкретный конечный результат в виде капитального строения. Или какие-то действия нацеленные на социально - экономическое совершенствование муниципального образования города Благовещенска посредством разработки мероприятий муниципальной программы.

Задачей является – объединение взаимоувязанных программных мероприятий, как правило, нацеленных на достижение совершенно точно определенных целей реализации муниципальной программы.

Программное мероприятие муниципальной программы (подпрограммы) – совокупность определённых взаимосвязанных мероприятий или определенных действий, направленных на решение конкретной задачи определённой муниципальной программой (подпрограммой).

Целевой показатель – показатель, элемент программы, позволяющий в количественном и качественном выражении оценить результат достижения конкретно определенных целей. Под эффективностью муниципальной программы понимают процентное отношение достигнутых результатов к разнообразным ресурсам, затраченных на их достижение. Муниципальная программа имеет несколько состояний, которые меняются по мере совершения определённых действий или другими словами Жизненный цикл. Состояния муниципальной программы представляют собой последовательные действия, которые следуют в определённом порядке. Жизненный цикл муниципальной программы представлен на рисунке 4

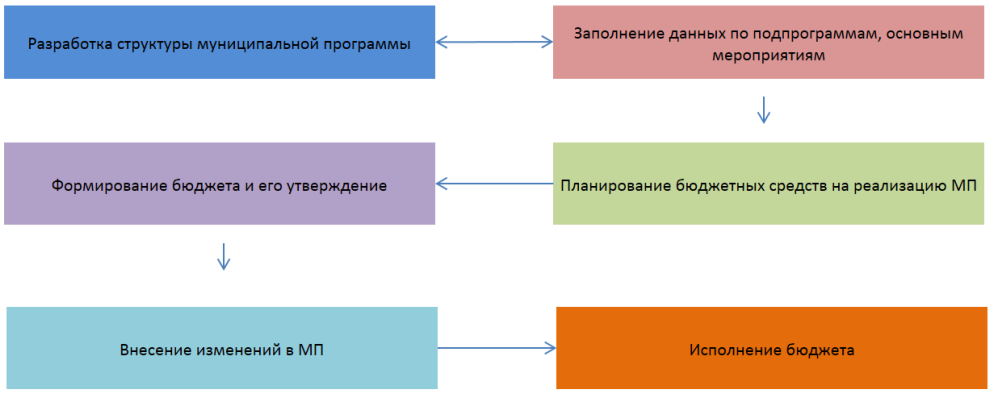


Рисунок 4 – Жизненный цикл муниципальной программы.

Учитывая многогранность структуры, попробуем разобраться в бизнес - процессах. Бизнес-логика – отражает совокупность правил, критериев работы, обусловленностей меду объектами в предметной области; применение правил и ограничений автоматизируемых и оптимизируемых операций. В рассматриваемом случае, предметной областью являются процессы от начала разработки своей Муниципальной программы до её конечной реализации. В процессе бизнес-моделирования и проектного конструирования по правилам протекающих бизнес-процессов, как правило, схема, описываться в виде:

- текста;

- подробно детализированных моделей предметной области.

На стадии анализа и проектирования информационной системы бизнес - поток представляется нам в различных диаграммах языка UML или ему подобных. На стадии программирования бизнес - логика изображается в коде обособленных сущностей, классов и их методов с инициализацией полей, при использовании современных объектно-ориентированных языков (ООП) программирования [1]. Стоит отметить, что бизнес-логикой также могут называться программные модули, её описывающие и исполняющие. В многоступенчатых (многослойных) информационных системах этот слой ассоциируется с нижележащим слоем подсистемы, также слой инфраструктурных сервисов, контроллером доступа к СУБД и подчинённым слоем приложения, который, функционирует со слоем графического клиентского интерфейса или наружными системами.

Apache – набор серверов для нескольких служб Интернета распространяется как пакет для установки на операционные системы Linux

1.2.3 Постановка задачи

Целью разрабатываемой информационной системы является повышение эффективности работы администрации города Благовещенска. Для достижения этой цели нужно провести комплексную автоматизацию всего процесса исполнения бюджета. На данном этапе модуль программного обеспечения должен хранить и выдавать информацию в информативной форме. Для этого необходимо провести ряд мероприятий:

- проанализировать обмен информацией между структурными подразделениями;

- упростить получение новых данных корректировки бюджета;

- устранить дублирование информации;

- автоматизировать формирование отчётных форм.

Это программное обеспечение позволит сократить ручные операции, ускорить обработку информации, ускорить реакцию на изменения бюджета,

позволит оперативно принимать управленческие решения. Руководство администрации в любой момент времени будет иметь необходимую оперативную информацию по реализации муниципальных программ. Немаловажным фактором повышения эффективности работы является организация строгого контроля исполнения мероприятий.

Входной информацией для создания информационной подсистемы должно служить технологическое задание, которое в свою очередь формируется в соответствие со стандартом ГОСТ 34. 602 - 89 «Техническое задание на создание автоматизированной системы». Проработка технического задания осуществляется во взаимодействие с заказчиком информационной системы и на основе данных, полученных в ходе анализа предметной области. Техническое задание на разработку представлено в приложении А. Оно, включает в себя:

- общие положения, представляющие собой основные наименования участвующих сторон (заказчика, исполнителя);

- предназначение и цели внедрения системы. Они определяют его ожидания от создания и внедрения системы;

- требования к системе, определяют то, какую систему хочет получить заказчик в результате проведения работ по ее выстраиванию;

- состав и содержание работ по воссозданию системы, регламентируют различные технико-экономические аспекты и временные затраты на воссоздание системы;

- порядок госконтроля и приемки системы, определяет набор документов и действий по сдаче системы заказчику и ее утверждению;

- предписание к составу и содержанию работ по проработке предмета автоматизации к запуску системы в действие, устанавливает последовательность действий необходимых для ввода системы в эксплуатацию;

- предписания к документированию, состав комплекта документов, образующийся в ходе проектирования и разработки данной системы;

- источники разработки, описываются различные информационные ресурсы, послужившие основой для разработки.

Для выполнения выпускной работы необходимо, в соответствия с требованиями заказчика, разработать программное обеспечение, которое реализует оговоренные с заказчиком функции. В соответствии с техническим заданием, выделим задачи, которые должна решать система:

- однозначное определение пользователя в системе;

- сбор и обработка информации о пользователях;

- предоставление пользователю возможности добавлять, изменять и удалять внесенную им информацию;

- разделение функций для различных ролей пользователей;

- проверка вводимых пользователем данных и предотвращение не - преднамеренных или злонамеренных воздействий;

- при возникновении внештатных ситуаций или сбоев предоставлять пользователю информационные сообщения со способами разрешения таких ситуаций и контактной информацией службы поддержки;

- сбор и обработка информации о работе самого приложения. Выходной информацией должны служить сведения о прохождении системой этапов, определенных порядком контроля и приемки системы, представленным техническом задании, по следующим пунктам:

- анализ готовой системы;

- сравнение разработанной системы с техническим заданием на ее разработку, с целью определения выполнения всех предъявленных в нем требований;

- выполнение доработки и изменений системы при необходимости; - опытная эксплуатация системы в режиме бета -тестирования;

- доработка системы и исправление ошибок. Постановка задачи разработки четко определяет вектор действий по проектированию и дальнейшей разработке приложения. Действия, произведенные на данном этапе, позволяют сформировать видение предметной области, определить цели и необходимость создания системы, а также в общих чертах ознакомиться с требованиями заказчика.

1.3 Анализ и сравнение существующих технологий для проработки web-приложений

В современном мире у проектировщиков web-приложений есть разнообразнейший выбор языков проработки программного обеспечения и возникает вопрос какой всё-таки применить для написания приложения. Вариантов великое множество. Самыми распространёнными среди программистов сейчас ультра ультрасовременная платформа PHP и ASP. Net.

1.3.1 Сравнительный анализ платформ кодирования

Пакет программ PHP и ASP. Net PHP – язык для составления сценариев серверного слоя (алгоритмов). Интерпретатор этого языка совершенно общедоступен, с вовсе свободным базовым кодом, для распространения его были воссозданы концептуальные сборки для всевозможных веб-серверов в первую очередь для хост-сервера Apache и IIS. В настоящее время этим системным продуктом заинтересовывается корпорация Microsoft. Началось эффективное сотрудничество с разработчиком Zend, в итоге чего появилась долгожданная встроенная усиленная поддержка модулей и самого PHP в IIS [15].

ASP. Net это неоднородная развивающаяся технология программирования, не характеризующаяся мощным языком. Постулат работы ASP. Net приложения чрезвычайно сильно различается от прошлой версии ASP. Разницу проанализируем далее [11].

PHP является языком web-программирования, который разрешает динамически выдавать HTML разметку или диалоговую полезную информацию в виде дифференцированных данных так, кстати подзаголовки для Cookie. Например:

<? php print “Диаграмма”;? >

PHP - код можно вставлять в типовую статичную HTML разметку:

<? php print “Отчёт”;? >.

Специфическим отличием для ASP. Net Webforms относительно PHP является иная реализация постулата работы, где каждая новая страница сводится из трех файлов: файла с HTML-разбивкой, ASP. Net контролами и файла сер-

верного слоя. Всеобщая сущность метода состоит в разделении на слои типа наружного вида страницы от её алгоритма логичности: файл разбивки будет содержать только разбивку (описывать наружный вид визуальной формы), а файл логичности только сухой программный код. Способность динамически подкидывать программный код в файл гипертекстовой разбивки и в файле бизнес-логичности генерировать HTML на лету разметку не запрещается, но это уже будет подход, аналогичный PHP. В гипертекстовой разметки страницы обязательно есть хэштеги, которые не характерны для HTML разметки, такие теги как asp: TextBox будут трансформированы. Серверным контролам можно задать состояние типа размер, добавить JavaScript и т. д. Соответствующий файл с логикой в таком случае имеет вид: namespace TestWebApplication;

Все порождённые на странице серверные контролы легкодоступны в файле с бизнес-логичностью по назначенному им коду. К примеру, выше программно реализуется доступ к элементу Client и задается текст, который он будет отражать. Действия по обновления состояния объекта обеспечиваются на этапе переработки страницы сайтом изданий (IIS), далее у среднестатистического пользователя в браузер отрисовывается готовая HTML разметка, не имеющая серверных компонентов [19].

Даже по объёму приведённых приблизительных текстов для каждой из технологий видно то, что критерии работы PHP элементарнее, нежели ASP. Net. Это является одной из главных первопричин широкого распространения PHP. В различие от PHP, ASP. Net потуги использовать при конструировании Web-приложений те же основополагающие постулаты, что и при реализации Windows-приложений. Протокол HTTP протокол, разрешающий контактировать приложениям продолжительное время между собой по критерию вопрос ответ. Так как Windows-приложения как правило функционируют непрерывно для этого нужно использование множества абстрактностей, приспособленных маскировать «сеансовость» протокола HTTP. Положительным свойством их является то, что существенно упрощается механизм разработки Интернет-приложения, и вследствие чего разработка приложения занимает минимальное

время [11].

Сам по себе ООП язык PHP выступает в роли Си-подобного без какой-либо типизация полей переменных, но с поддержкой объектно-ориентированного метода. При создании разнообразнейших web-приложений ASP. Net целесообразно применять абсолютно любой язык из совместимой с ним платформы. Net C #. Net. Доступна вся функциональность данных библиотек линейки. Net Framework, что значительно понижает преимущество PHP. Во-первых, по возможностям PHP отстает от DOT Net. К примеру в PHP5 была намного улучшена поддержка ООП, но она не соответствует модели. Net, где языки кодирования такие как C # и VB. Net характеризуются всецело ООП. PHP язык очень распространен среди разработчиков программного обеспечения в первую очередь используется для быстрой разработки сайтов в сети Интернет. По мимо этого он является полноценным языком обладающим принципом ООП. На этом языке можно создавать абсолютно любые приложения. На пример можно обрабатывать большие массивы данных подключаясь к серверу баз данных Oracle.

Безусловно, по документированным возможностям интерпретатор PHP отстает от DOT Net. К примеру, в PHP5 была чрезвычайно сильно усовершенствована поддержка ООП, но она не соответствует модели. Net, где языки кодирования такие как C # являются ООП изначально.

В объектно-ориентированных языках очень сложно создать переменную, а только элемент класса, нельзя создать функцию – только как метод родительского класса. В PHP, нет событий (развёртывание событий основывается на применении внешних способов, например, через добавление функции в свойства тега, воссоединение AddEventListener к объекту, но это приводит к проблемам из-за огромного числа всевозможных браузеров), не существует способности перегружать ранее созданные методы (вследствие неимения строгой типизации в языке) – это маленькая часть тонкостей, которые появляются при задействовании PHP относительно настоящих объектно-ориентированных языков. PHP имеет все превосходства функционального кодирования к примеру

функции высшего порядка, простые функции.

В PHP не реализована многопоточность, а также не поддерживаются или некорректно работают некоторые кодировки, к примеру, Unicode. PHP является наименее строгим языком, нежели. Net -языки, из-за полного отсутствия строгой типизации переменных и отсутствия принудительного объявления переменных. В результате чего будут воссозданы разные небезопасные оснастки, которые не будут приняты компиляторам и попросту не скомпилируются даже с помощью мощного C, но при этом в PHP выполняться с неожиданным результатом.

В свою очередь не обходимо отметить, что в языках. Net программа завсегда тщательно проверяется компилятором на допущенные разработчиком синтаксические ошибки. Для платформы PHP имеется обилие сред упрощенной разработки например широко известная: Zend Studio для PHP, или недавно выпущенный в свет плагин под Visual Studio. Все приведенные выше среды довольно неслабо развиты и широко используются кодерами в процессе программирования специалистами по всему земному шару что связано с поддержкой великого множества современных возможностей сред обеспечивающих облегченную разработку.

На сегодняшний день самый современный и пожалуй популярный коммерческий продукт, который применяется для разработки web-приложений на ASP. Net – это могучий Microsoft Visual Studio. Есть и посторонние средства проектирования и разработки к примеру, Visual Code, но они наименее распространены.

По большому счету, все программные продукты для разных платформ кодирования имеют подключённую систему интеллектуального помощника кодирования (IntelHsense), возможность допуска через специальный драйвер к базе данных, способность визуальной высоко интелектуальной отладки. Впрочем некоторые различия все же существуют. В разработке абсолютного большинства приложений очень сильно помогает IntelHsense при умелом ее применении. Но в PHP эта идея не может быть так широко использована, как в ASP. Net

снова из-за того, что он всё таки не строго типизированный язык. В большинстве моментов вместе с PHP используется СУБД MySql. А для ASP. Net-приложения применяют уровень сервер баз данных так например для Microsoft – Microsoft SQL Server. Две похожих технологии умеют функционировать с любыми СУБД независимо от производителя. Если сравнить MySQL и Microsoft SQL Server, то здесь с громадным перевесом побеждает сервер от Microsoft. И одна специфика MySQL (при использовании совместно с PHP) – возможно на ряду с классическими приложениями такими как для Windows использование доступ через браузер к базе данных. В качестве визуального интерфейса для MSSql надо применить SSMS, поставляемую вместе с ядром базы данных, либо попользоваться посторонними инструментами к примеру, DbForge.

1.3.2 Сравнительный анализ MSSQL и MySQL

SQL – язык запросов, грубо говоря язык на котором сочиняют запросы к базе данных. Именно с его помощью обеспечивается выборка и преобразование информации. В течении 30 лет он трактуется самым популярнейшим нормативом в этой сфере. За этот промежуток возникло большое количество структур управления базами данных, выстроенных на основе SQL, – так именуется программное обеспечение, которое применяет этот язык. MySQL одна из таких СУБД [35].

В настоящее время MySQL является одним из самых популярнейших средств управленья базой данных, хотя у него наличествует много соперников. Например, Microsoft MS SQL Server. И если говорить о сопоставлении, то будет уместно определить возможности этих двух серверов баз данных. Что представляет собой MS SQL Server Эта СУБД стала первой реализацией Microsoft, нацеленной на коммерческое, а не домашнее применение.

Возрастание популярности клиент-компьютерной обработки данных оживило в руководстве интерес к этой нише. Первая интерпретация вышла в 1988 году, была коллективным проектом с компанией Sybase и приобрела достойную оценку прессы. В свойстве основного языка запросов применяется

особенное процедурное расширение типового SQL, получившее наименование Transact-SQL [35].

MySQL эта СУБД поддерживается фирмой Oracle, что делает ее непосредственной продолжательницей первых систем, построенных на SQL. Первая трактовка вышла в 1995 году. Это программное обеспечение приобретает статус независимого, то есть любой пользователь вправе не только безвозмездно скачать и установить ее, но и привнести собственные преобразования в ее код, который общедоступен для публики [35].

Общие черты двух СУБД Если отступить от того, что обе СУБД основываются на одном языке программирования, то общего у них не так много. Две системы трактуются известными продуктами с продолжительной историей и солидной поддержкой у SQL Server централизованной, у MySQL учреждённой на необъятном содружестве профессионалов. У обоих приблизительно одинаково высокий показатель безопасности. Зато расхождений гораздо больше, чем на первый взгляд. Расхождения [28]:

Сертификационный статус. Как большинство товаров Microsoft, расширенная версия SQL Server коммерческая. Впрочем существует вариант для общедоступного распространения. Его различие от расширенного продукта состоит в приставке Express и некоторых существенных ограничениях в работе. MySQL же не только общедоступна, но и имеет лицензию свободного ПО. Любой желающий может снискать доступ к ее исходному коду.

Операционная система. До недавнего времени MS SQL функционировала исключительно в операционной среде Windows. Поддержка Linux была реализована только в версии 2017. Для MySQL изначально проектировалось для Linux по этому не имеет принципиального значения в какой операционной системе ей работать, она с одинаковым успехом устанавливается и на Windows, и на UNIX. То же можно сказать о средствах реализации SQL кода, применяемых в связке с этими СУБД. MS SQL прекрасно сочетается с площадкой. NET, а MySQL – популярнейшая система у веб-разработчиков, занимающихся реализацией на PHP и поддержкой сайтов на Linux.

Синтаксис. Невзирая на общую концепцию, написание одного и того же запроса в двух различных СУБД может коррелировать. Системные требования. MS SQL Server требуется в огромные мощности. По этой причине она чаще применяется для бизнес решений, а MySQL для компактных web-приложений. Совершенствование функциональности. MS SQL Server (разумеется, в платной версии) обладает наиболее обширными возможностями уже в основополагающей комплектации. В систему входят инструментарии анализа данных, сервер аналитических протоколов и иное эффективное программное обеспечение, чего не проговоришь об MySQL при его задействовании придётся разыскивать сторонние решения. Впрочем это привносит MySQL несомненно большую пластичность по сравнению с самодостаточностью продукта Microsoft.

Простота применения. Нахождение всего нужного для удобной работы в одной сборке, привычная установки с помощью мастера установки схожа мастеру установки любого программного обеспечения в Windows и безупречное функционирование с другими продуктами Microsoft делают SQL Server структурой, очевидной даже малоопытному программисту. Для отладки MySQL и поиска максимально подходящих расширений придется потратить время. И MS SQL Server, и MySQL не зря снискали свою востребованность. Обе СУБД обладают бесспорными преимуществами, а их специфичности позволяют учитывать способности отдельно взятого сервера и специалиста, который специализируется поддержкой базы данных [28].

Выбор определённой СУБД характеризует собой трудную многопараметрическую супер-задачу и является одним из немаловажных этапов в повседневной разработке информационной системы. Выбранный разработчиком программный продукт непременно должен максимально сильно удовлетворять всем нынешним, и даже грядущим востребованности которые ещё нужно предсказать, при этом следует учитывать финансово-экономические траты на приобретение целесообразного серверного оборудования, самой системы СУБД, разработку желательного программного обеспечения на ее структуре, а также обучение контингента. Очевидно, что весьма простой метод при выборе серве-

ра баз данных построен на характеристике того, в какой степени существующие системы удовлетворяют главным требованиям формируемого проекта информационной системы. Наиболее трудным и дорогостоящим сценарием характеризуется параллельное создание демонстрационного проекта базирующегося двух-трёх СУБД и дальнейший выбор максимально подходящего из представленных кандидатов. Однако и в этом прецеденте целесообразно ограничивать по максимуму круг вероятных систем, принимая во внимание некие выработанные профессионалами своего дела критерии отбора.

Тождества между рассматриваемыми СУБД SQL и MySQL [28]:

- Обе имеют возможность работы с реляционной базой данных;

- Обе характеризуют тип данных;

- Обе применяют индексы как ключевые поля, хранимые процедуры, представления;

- SQL рассматривается основополагающим языком для MySQL;

- Обе применяют для работы арифметической операции (+, -, *, /, %);

- И та и другая, может выполнять операции сравнения (>, <,> =, <=);

- Могут осуществлять логические операции (и, или нет);

- Содержат коды идентификаторы для формирования взаимосвязей между разными по структуре и содержанию таблицами одной базы данных (базисный ключ, наружный ключ);

- Способны применить алиазы;

- Могут консолидировать разные таблице по логике и содержимому при наличии объединяющего ключа в самой таблице (внутреннее сопряжение, внешнее сопряжение, левое сопряжение, правое сопряжение);

- Могут задействовать агрегатные структуры В чем собственно различия между двумя столь схожими понятиями SQL и MySQL.

SQL – это язык систематизированных обращения к какой либо базе структурированных данных для полного управления этими данными проще говоря для управления содержащимися в ней структурированными данными.

MySQL – это концепция управления самими реляционными базами данных которые в свою очередь используются для долгосрочного хранения полезных данных, извлечения информации по запросу, видоизменения и администрирования базы данных с задействованием SQL [8].

Компоновка базы данных: SQL это собственно язык структурированных особым образом запросов, MySQL обеспечивает консолидированную среду инструментариев «MySQL Workbench» для конструирования и программирования баз данных.

Соединители: SQL как повелось изначально не предоставляет, однако есть альтернатива соединителей MySQL обеспечивает драйверы баз данных для площадки. NET, PHP для воссоздания подсоединения кода к базе данных. Основополагающее противопоставление между SQL и MySQL сводится в том, что SQL это всё таки язык запросов для всестороннего расширенного управления всеми расположенными данными в реляционной базе данных, а MySQL это подсистема управления реляционными базами данных с открытым исходным кодом для управления самими базами данных с помощью SQL [30].

1.4 Анализ шаблонов проектирования

Шаблон проектирования или разработки современного программного обеспечения на языке высокого уровня представляет заранее определенный метод создания структурированного программного кода предназначенного для упрощенного алгоритма действий для решения конкретных, довольно часто встречающихся, повторяющихся задач разработки и предварительного проектирования информационных систем.

Шаблоны разработки, чаще всего, не зависят от конкретного языка высокоуровневого программирования. А так же их фундаментальные принципы и методы применения будут идентичны и в C #, и в PHP, и в иных объектно-ориентированных современных языках кодирования. Также шаблоны упрощают коллективную разработку кода информационной системы. Предполагается, что всегда есть некоторый стандартный набор определенных часто востребо-

ванных логически формализованных задач и шаблоны предлагают разработчику широкий спектр алгоритмов для их разрешения.

При разработке авторского программного кода для создания программного обеспечения применение шаблонов дает возможность формализовать имеющуюся проблему в виде набора конструкторов, классов, программных объектов предполагаемых и существующих взаимосвязей между ними. Затем применяется один из приемлемых шаблонов для решения этой задачи и используется готовый набор правил. Зная заранее применяемый шаблон проектирования современного программного обеспечения и его основополагающие постулаты работы другому разработчику будет гораздо проще понять его логику вносить свои доработки при модернизации системы. В первооснову типологии наиболее распространенных алгоритмов положена сверхзадача или сверхзадачи, которые этот шаблон решает.

Порождающие шаблоны отделяют процесс создания (порождения) дочерних классов и объектов. Среди таких объектов выделяют патерны [24]:

«абстрактная фабрика» (Abstract Factory);

«строитель» (Builder);

«фабричный метод» (Factory Method);

«прототип» (Prototype);

«одиночка» (Singleton).

Функциональные паттерны проектирования рассматривают как классы, и модули которые формируют сверхкрупные подсистемы наиболее запутанные по характеру классы и модули. К таким паттернам относятся [24]:

«адаптер» (Adapter);

«мост» (Bridge);

Поведенческие разработанные шаблоны определяют последовательные алгоритмы взаимодействия между классами и объектами [24].

Среди подобных паттернов можно выделить:

«цепочка обязанностей» (Chain of responsibility);

«команда» (Command);

«наблюдатель» (Observer).

Шаблонов проектирования программного обеспечения гораздо больше приведенных выше паттернов. При выборе наилучшего шаблона необходимо тщательно выделить все существующие абстрактные сущности и связи между этими объектами, а так же изолировать их от конкретной сущности. После этого следует выявить, вписывается ли данная абстрактная модель решения поставленной задачи в некоторый шаблон. Например, при создании новых экземпляров объектов наилучшими могут оказаться порождающие модели.

При уверенном использовании описанных шаблонов для проектирования, как и при любом другом стиле разработки собственного программного обеспечения, следует придерживаться основного и наиболее правильного принципа KISS (Keep It Simple, Stupid) сохранять весь код своей программы максимально простым и понятным оставлять комментарии по коду. Смысл всех существующих паттернов состоит не в усложнении процесса написания программного кода, а в его упрощении.

1.4.1 Паттерн «Наблюдатель» (Observer)

Назначение паттерна или шаблона проектирования заключается в определении существующих зависимости типа «один ко многим» между формализованными объектами так, что при изменении текущего состояния одного единственного объекта все зависящие от него объекты оповещаются об этом и автоматически обновляются. Наблюдатель уведомляет все заинтересованные стороны о произошедшем событии или об изменении своего состояния [24].

Существует два способа общения между двумя программными элементами. Компонент системы 1 может обратиться к Компоненту 2 этой же системы для получения каких-то важных данных или выполнения запрограммированной операции. В этом случае Компонент 2 информационной системы или модуля выполняет определённые действия, когда придёт определённый сигнал. В некоторых случаях Компонент 2 является активным, содержит собственный поток исполнения или каким-то другим способом постоянно следит за своим состоя-

нием. В этом случае Компонент 2 может уведомить Компонент 1 о случившемся событии.

Первая модель взаимодействия называется риП-моделью, а вторая push-моделью. Push-модель взаимодействия объектов системы появилась задолго до распространения ООП и паттернов проектирования высокоуровневых языков. В мире структурного программирования push-модель выполняется с помощью метода обратного вызова (callbacks). В функциональном программировании push-модель представлена в виде реактивной модели. При использовании современного объектно-ориентированного принципа push-модель строится при помощи шаблона разработки «Наблюдатель».

1.4.2 Паттерн «Одиночка» (Singleton)

В автоматизированной структуре одно моментно могут функционировать разные сущности в одиночном виде (автоматизированная подсистема ведения журнала уведомлений системы). В таких случаях потребуются навыки умения воссоздать один-единственный уникальный экспонат определённого типа, пре доставлять к нему ресурс из другого экспоната класса и воспрещать создание иных инстанциированных экземпляров одного и того же типа. Для программирования таких целей служит паттерн Singleton.

Структурная схема паттерна Singleton построена на концепции задействования глобальной строго типизированной (при возможности) переменной, имеющей необходимые свойства для чтения и установки новых значений [24].

Такая переменная (глобальная) общедоступна постоянно (время жизни такой переменной от старта программы до полного ее завершения). Эта переменная, как правило, может быть общедоступна из любой части нашей программы. Разумеется, задействовать такую переменную неопределённого типа напрямую сложно, так как существует трудность предоставления единственности экземпляра. Для решения этой проблемы существует разработанный специалистами паттерн Singleton гарантирует непрекращающийся контроль над созданием одного-единственного объекта через родительский класс. Доступ к объекту обеспечивается через статическую функцию одного-единственного член класса, которая возвращает только указатель или ссылку на него. Этот

объект будет мгновенно создан при первой итерации к вызываемому классу или публичному методу, все дальнейшие вызовы этого объекта будут возвращать его адрес в оперативной памяти. Для обеспечения безусловной неповторимости вновь созданного объекта или интерпретативной сущности, конструкторы и операторы трансплантации комбинирования всегда объявляются приватными, то есть, закрыты для обращения на прямую, что бы получить или записать в них значение необходимо получить доступ через соответствующее свойство.

Такая схема доступа к элементу класса в данном случае полю через его свойство, необходимо для обеспечения нормы по требованию безлопастного программирования. Эти требования рекомендованы компанией Microsoft для разработчиков программного обеспечения.

1.4.3 Паттерн «Абстрактная фабрика» (Abstract Factory)

На сегодняшний день известны две зарекомендовавшие себя уже ставшие классическими разновидности шаблонов разработки: «Абстрактная фабрика» и «Фабричный метод». Обе разновидности представленных штампов проектирования предназначены для инкапсуляции созданного объекта или семейства объектов. На практике очень часто отходят от классических реализаций паттернов и называют фабрикой любой класс, инкапсулирующий в себе воссоздание объектов.

Абстрактная фабрика обеспечивает интерфейс для воссоздания подсемейства взаимозависимых или неразрывно связанных объектов, не специфицируя их определённых классов. Иными словами, абстрактная фабрика являет собой методологию создания подсемейства взаимозависимых или неразрывных объектов [24].

Архитектурный метод, при котором формулировано прямое функционирование компонент- компонентов считается никудышной практикой и приведет к недоработкам. Концепция паттерна «наблюдатель» разрешает синхронизировать изменения зависящих компонентов. Предложенная методология позволит избежать неопределённости при приросте графических элементов пользовательского интерфейса.

1.5 Анализ существующего программного обеспечения

На рисунке 5 приведено ПО Министерства экономического развития Амурской области. Разработанное ПО решает задачи мониторинга программных мероприятий в полной мере, однако, графический интерфейс этого ПО крайне сложен, не информативен, тяжёл к запоминанию. Полностью отсутствуют наглядные промежуточные отчёты. На рисунке 6 приведен пример WEB интерфейса программного комплекса «Государственные программы» разработан для автоматизации государственного и муниципального планирования, мониторинга и анализа эффективности реализации государственных программ. Программный комплекс разработан НПО Криста. В таблице 1 приведены сравнительные характеристики ПО, спроектированного для решения поставленной задачи.

Наименование / Параметры	ПО министерства экономического развития Амурской области	ПО «Государственные программы»	Проектируемое ПО
Соответствие НПА	да	да	да
Эргономичный интерфейс пользователя	нет	да	да
Дополнительные отчётные формы	нет	на заказ	да
Промежуточные отчёты	нет	на заказ	да
Подготовка доклада мэру	нет	нет	да
Техническое сопровождение	нет	1 год бесплатно	постоянно бесплатно
Платно\бесплатно	бесплатно	платно	бесплатно

Таблица 1 – Сравнительные характеристики программного обеспечения

Как видим из таблицы 1 выбор варианта с разработкой собственного программного обеспечения наиболее логичен.

Достижение поставленной цели было разбито на этапы:

- исследование предметной области;

- подготовка технического задания;

- проектирование и разработка программного обеспечения;

- ввод в опытную эксплуатацию (включая обучение пользователей);

- исправление недочётов;

- ввод в продуктивную эксплуатацию.

На рисунке 5 изображено программное обеспечение министерства экономического развития Амурской области, интерфейс программного обеспечения полностью текстовой. Такое техническое решение имеет определённый смысл только, в случае если ощущается острая производственная необходимость уменьшения нагрузки на серверную составляющую разрабатываемого программного обеспечения или разработанной информационной системы, то есть при таком исполнении графического интерфейса значительно снижается нагрузка на сервер. Но в то же время отсутствует хоть какое-либо выделение иерархии объектов (шрифт, цвет), в результате чего интерфейс становится запутанным, что на мой взгляд сильно усложняет работу с информационной системой. Интерфейс программного обеспечения НПО Криста намного дружелюбнее.

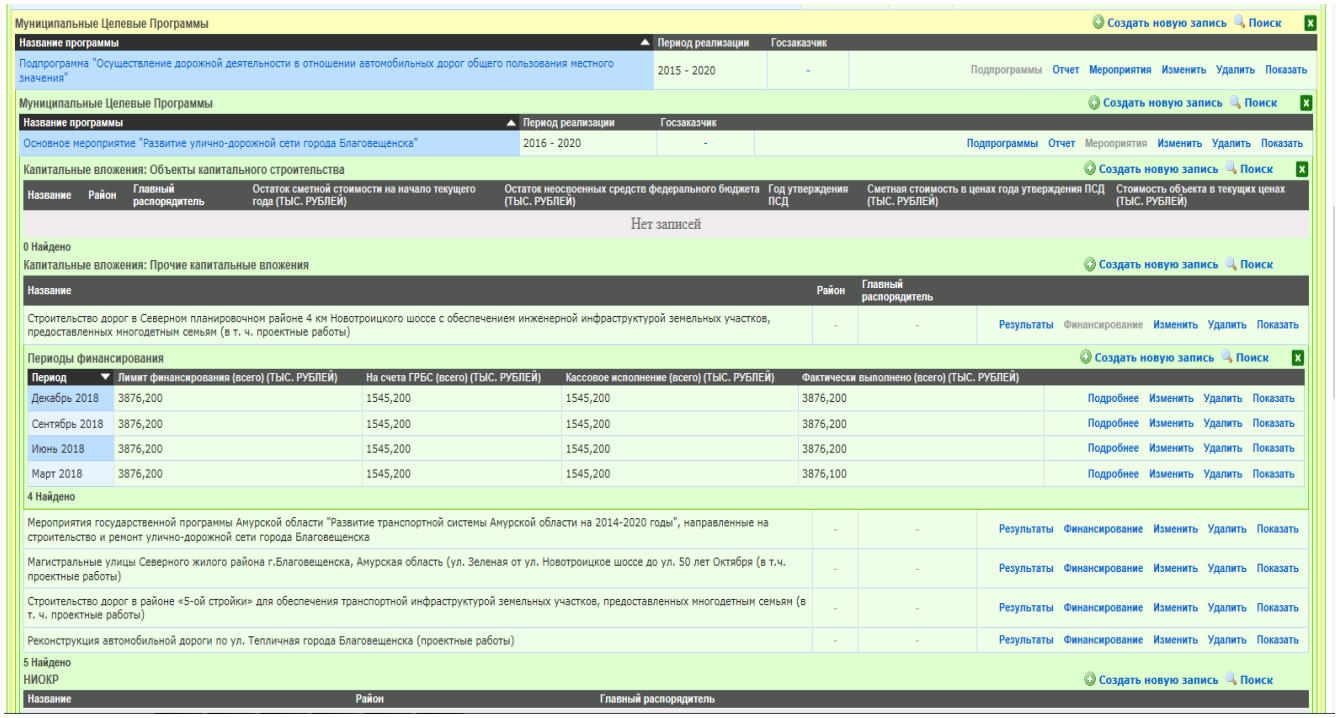


Рисунок 5 – Программное обеспечение министерства экономического развития Амурской области

В целом программный комплекс выполняет поставленные задачи, имеет разнообразные формы вывода информации: графики, диаграммы, таблицы. Имеет механизм загрузки и выгрузки (импорт, экспорт) данных. Минусом комплекса является его стоимость, отсутствие технической информации в публичном пространстве.

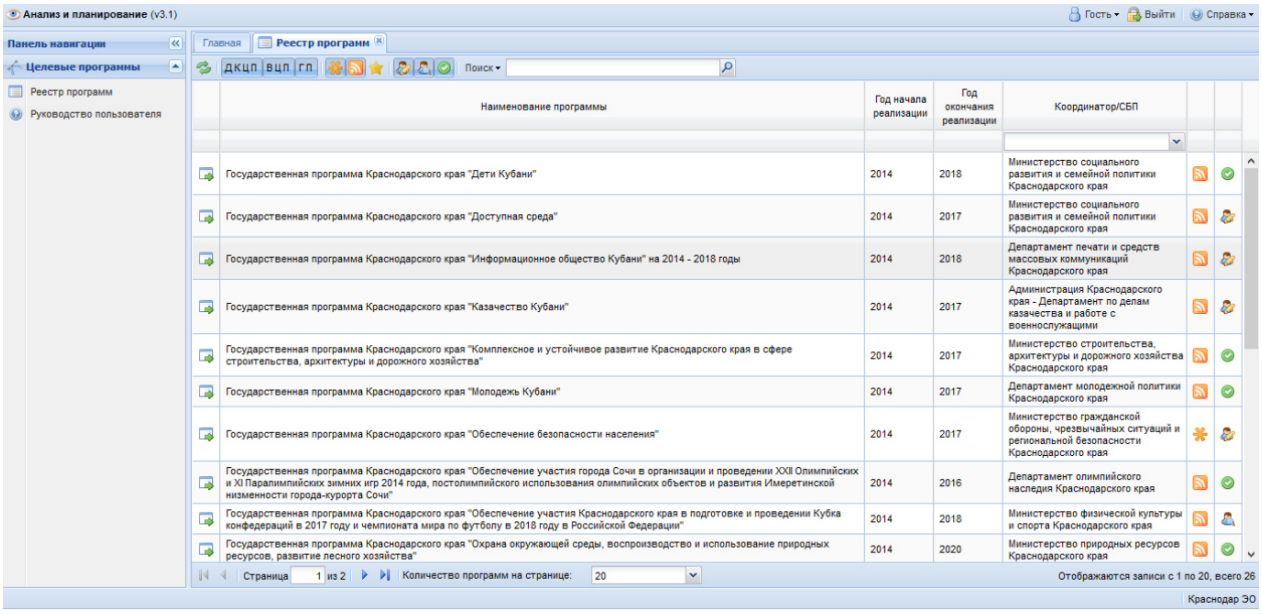


Рисунок 6 – Пример WEB интерфейса программного комплекса «Государственные программы»

Учитывая острую нехватку денежных средств, администрация города экономит, в том числе на закупаемом ПО, что заставило отказаться от него.

1.6 Выводы по главе

Эффективным методом повышения производительности труда является применение автоматизированной системы, которая способствует увеличению производительности труда, что благоприятно сказывается на исполнении бюджета. Основным документом в этой системе является финансирование мероприятия, представляющее информационно-динамическую модель, в которой отражаются все логические взаимосвязи и результаты выполняемых программных мероприятий, необходимых для достижения конечной. Для каждого мероприятия своя цель, но все вместе мероприятия формируют общую картину исполнения плановых мероприятий и собственно исполнение бюджета. Все мероприятия представлены с клиентской стороны, поэтому в традиционной архи-

тектуре типа «клиент-сервер» доводится определять три основополагающие части приложения по двум физическим модулям. Традиционно ПО складирования данных размещается на сервере (например, сервере базы данных), интерфейс с пользователем на стороне клиента, а вот переработку данных доводится переслать серверной части. На сегодняшний день у проектировщиков веб-приложений есть богатейший выбор относительно того, какой язык (или технологию) задействовать для создания приложения. Сценариев много: Perl, PHP, ASP, ASP. Net, JSP. Самыми распространенными языками объектно-ориентированного программирования (с серьёзнейшим отрывом от прочих) на нынешний день бесспорные лидеры PHP и ASP. Net. В результате тщательного изучения предметной области и различных широко используемых технологий реализации информационных систем с доступом по протоколу HTTP или Web-протоколу было решено:

1) Выбрать в качестве технологии реализации платформу PHP, в связи тем, что PHP ориентирована как на большие проекты, так и на более мелкие проекты. В PHP код не нуждается в компиляции за счёт чего выполнение его происходит невероятно быстро. PHP возможно развернуть в операционных системах семейства Linux. Что в свою очередь существенно снизит финансовые затраты.

2) Наиболее подходящим шаблоном проектированием является Observer (Наблюдатель) в связи необходимостью постоянного контроля над длительностями на диаграмме Ганта.

3) Среди существующего программного обеспечения наиболее приятным для восприятия пользователей определено коммерческое программное обеспечение, однако для его внедрения и поддержки понадобятся финансовые вложения. Причём эти вложения будут постоянными в рамках технической поддержки программного обеспечении. Кроме этого при необходимости модернизации программного обеспечения придётся снова платить в то время как собственная разработка в финансировании технической поддержки и модернизации программного обеспечения не нуждается.

2 ФОРМИРОВАНИЕ ТРЕБОВАНИЙ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

При формировании технического задания в нем обязательно должны перечисляться все требования к информационной системе, так как разработчик должен знать, для какой цели создается продукт, а так же что он предназначен выполнять и каким образом. Задача формулирования требований ложится на заказчика, хотя на практике с этим обычно помогают менеджеры, через которых происходит оформление заказа [31].

2.1 Формирование требований к информационной системе

2.1.1. Назначение

Программное обеспечение «Мониторинг исполнения программных мероприятий администрации города Благовещенска» предназначено для ведения оперативного мониторинга исполнения муниципальных программных мероприятий, реализуемых на территории муниципального образования города Благовещенска.

2.1.2. Область действия

ПО имеет клиент-серверную архитектуру. Выполнено по принципу WEB технологии. ПО собирает, хранит и представляет данные в виде отчётов, графиков, диаграмм. Данные вводятся ответственными исполнителями в части исполняемых мероприятий.

2.1.3. Определения, акронимы и сокращения

Администратор (Заказчик) – управление экономического развития администрации города Благовещенска;

Ответственный исполнитель – участник программного мероприятия, распорядитель бюджетных средств;

Разработчик ПО – управление ЕМИС;

Технический специалист – специалист управления ЕМИС, осуществляющий техническое сопровождение ПО;

Отчёт – вывод информации в табличном виде (по ранее представленным формам);

График, Диаграмма – вывод обобщенной информации в соответствии с алгоритмом Администратора;

Интерфейс пользователя – графический интерфейс пользователя разрабатывается в соответствии с эскизами Администратора.

2.1.4. Ссылки

Послание Президента РФ;

Бюджетный кодекс;

Постановление администрации города Благовещенска.

2.1.5. Краткий обзор

ПО предназначено для осуществления оперативного управления бюджетными средствами, предусмотренными на реализацию программных мероприятий администрации города Благовещенска. Представляет собой клиент серверное приложение с WEB интерфейсом. Состоит из нескольких модулей:

модуль администратора – предназначен для аутоинтефикации и авторизации ответственных исполнителей;

модуль загрузки данных об исполнении бюджета из ИС финансового управления – предназначен для импорта данных об исполнении бюджета города Благовещенска из информационной системы финансового управления города Благовещенска;

модуль Ответственного исполнителя – представляет собой множество форм ввода и вывода данных, в том числе в виде отчётов, графиков, диаграмм. Данные подлежащие вводу в ПО – данные о названии, периодах проведения, финансировании муниципальных программ, подпрограмм, основных мероприятий, мероприятий, индикаторов и показателей. На основании введенных данных программное обеспечение формирует отчётные формы, графики и диаграммы. Позволяет руководителю получать оперативную информацию о реализации мероприятий в разрезе управлений, подразделений, исполнителей, об-

щую информацию об исполнении мероприятий или детализированную по конкретному мероприятию, исполнителю.

2.1.6. Общее описание

2.1.6.1. Взаимодействие продукта (с другими продуктами и компонентами)

Для функционирования серверной части ПО необходим MySQL и WEB сервер с подключенным модулем PHP 5. Для функционирования клиентской части необходим WEB Браузер с поддержкой стилей CSS [3]. Поддерживаемые браузеры: Internet Explorer, Яндекс браузер, Google Chrome. Операционная система: Windows 7 и более новая.

2.1.6.2. Функции продукта (краткое описание)

Наглядное представление информации и информирование о возможных проблемах с исполнением программных мероприятий.

2.1.6.3. Характеристики пользователя

Администратор ПО и Ответственный исполнитель должны иметь навыки работы с компьютером, иметь представление о структуре программных мероприятий администрации города Благовещенска. Уметь пользоваться Интернет браузером.

2.1.7. Ограничения

Вводимая информация должна соответствовать региональным стандартам.

Дата в формате: дд.мм.гггг.

Число в виде: 1,1 или 1.1.

Импортируемый файл, строго соответствующий структуре XML.

2.1.8. Допущения и зависимости

При изменении версии CSS необходима обновление ПО [20].

2.1.9. Детальные требования

2.1.9.1. Интерфейсы пользователя

Разрабатываются в соответствии с эскизами, предоставленными Администратором. Изменения вносятся по мере необходимости с предварительным согласованием Администратора и Разработчиком ПО. Текстовое поле ввода – любые символы, длина не более 2000 знаков. Числовые поля ввода – целочис-

ленные и с плавающей точкой разделенные запятой или точкой. Поля ввода даты - дд.мм.гггг.

2.1.9.2. Интерфейсы взаимодействия

В разрабатываемом программном обеспечении предусмотрена возможность загрузки файла данных о бюджете в формате XML

2.1.10. Функциональные требования

ПО должно обеспечивать доступность данных круглосуточно, обеспечивать восстановление информации на предыдущую дату.

Создавать резервные копии базы данных ежедневно.

ПО должно проводить проверку на корректность вводимых данных Ответственным исполнителем. При возможности проводить корректировку вводимых данных (исправлять опечатки).

2.1.11. Требования к производительности

Процессор: Dual core 3 ГГц.

Оперативная память: 2 ГБ.

Объем свободного места на диске 50 ГБ.

2.1.12. Проектные ограничения

IEEE Std 830-1998

2.1.13. Нефункциональные требования

Создание резервных копий базы данных ежедневно.

Доступность в ведомственной сети по IP адресу и порту 192.168.1.1:89

Безопасность данных осуществляется на уровне разграничения прав доступа пользователей.

На сервере необходима установка антивирусного программного обеспечения.

2.2 Моделирование прикладной области диаграмм в представлении IDEF0

Диаграмма классов IDEF0 предназначена, прежде всего для обозначения отношений между инстанциированными классами и их порождёнными экземплярами этих классов. Для чего-же это нужно? Это жизненно не обходимый

процесс при проектировании программного обеспечения, и не только для этого. Эта диаграмма подходит для непосредственного конструирования любой ситуации и может использоваться не только в прикладной области. IDEF0 по сути технология пошагового функционального моделирования происходит это название от англоязычного сочетания (англ. function modeling). Выполняется диаграмма в виде графической нотации, которая предназначена для обобщения и скрупулёзного описания протекающих событий. Присущей только этому виду диаграмм спецификой которых, прежде всего будет направленный нажим на жесткую иерархию строгой подчиненности сущностей. В IDEF0 иллюстрируется исключительно логические отношения между экземплярами классов или объектов [10].

Описание такого вида диаграмм выглядит как «чёрный ящик» со своими входящими сигналами, выходящими сигналами, управленьем бизнес - процессов и механизмом исполнения, после описания верхнего уровня он в свою очередь последовательно увеличивает детализацию до более низкого достаточного нам ранга. Для того чтобы правильно составить диаграмму такого типа целесообразно при изготовлении диаграммы пользовать специальные справочники которые содержат изложения всех возможных активностей и взаимосвязи с ними в виде стрелок. Справочники содержат подробные описания всех возможных взаимодействий. По этому не составит труда найти подробные изложения процессов того, какой смысл вкладывается в соответствующую активность, сущность либо указывающую стрелку. Описание принципа построения диаграммы IDEF0 содержится в рекомендациях Р 50. 1. 028 - 2001 Диаграммы А-0 и А0 приведены в приложениях Б и В.

2.3 Проектирование графического интерфейса

Неотъемлемой частью взаимодействия пользователя и информационной системы является графический или пользовательский интерфейс ещё называют WIMP -интерфейс [34].

Индексом простоты и удобства именуют Эргономикой (удобность использования) [33]. Графический интерфейс с высочайшим показателем эргономики характеризуется:

Изучаемостью. Из-за ожидаемо размещенным компонентам и знакомым канонам функционирования с ними, среднестатистический пользователь сравнительно без затруднений то есть быстро выучивает графические экранные формы системы, не прикладывая для изучения сверхчеловеческих усилий [13];

Эффективностью. Каждый заинтересованный сотрудник, взаимодействующий с системой, старается сделать конкретную свою работу быстро. Если сотруднику удалось быстро достичь поставленной цели и легко получить положительный результат, то скорее всего навигация системы и группировка содержимого отвечает установленным требованиям эргономики; Минимальным числом ошибок. Любая ошибка пользователя нарушает непрерывность наработки навыка и заставляет специалиста повторять выполненные ранее взаимодействия снова. Целиком избавиться от неточностей наедаться, так как заинтересованный сотрудник может снова сталкиваться с ними из-за собственных ошибочных действий или банальной невнимательности. Задача конструктора по возможности свести к абсолютному минимуму вероятность появления таких каверзных ошибок, а также предусмотреть их простое усовершенствование, например, в случае с неправильно заполненной формой;

Эстетичным дизайном. Параметр эстетики невероятно субъективен, впрочем, внешний вид системы должен коррелировать сложившимся стандартам. Уровень эргономики непосредственно связан с производительностью труда. Никудышная эргономика, бессознательно не понятный интерфейс неизменно связан с потерей времени сотрудников, что, в конечном счете, вытекает в финансовые потери. Системы с хорошим интерфейсом экономят существенные денежные средства. Когда заинтересованный сотрудник сможет решить подавляющее большинство своих повседневных задач в процессе работы с комфортным и функциональным интерфейсом, ему не придётся лишний раз взывать за помощью к проектировщику [23]. Простота пользовательского интерфейса бес-

ценное и, на сегодняшний день, наверное самое главное достоинство информационной системы. Однако, на самом деле всё обстоит не так, далеко не всегда удаётся применить свои знания и сделать программное обеспечение понятным и совсем несложным. Наряду с этим основой хорошей эргономики является беспрекословное соблюдение устоявшихся постулатов организации визуальных элементов и значков так заинтересованный сотрудник сумеет быстро найти интересующий его компонент в предсказуемом для него месте и при нажатии на иконку выполнится ожидаемое действие со стороны системы. Отдельно следует отметить следующие, на наш взгляд базовые постулаты эргономики системы. Фундаментальные принципы создания пользовательского графического интерфейса: Естественность (интуитивность). Работа с информационной системой ни в коем случае не должна порождать у обычных пользователей серьёзных трудностей с поиском нужных визуальных элементов оформления клиентского интерфейса для администрирования правилами приводящих к непосредственному решению поставленной задачи; Непротиворечивость. Если в процессе продолжительной работы с информационной системой ответственным специалистом были применены на практике определённые методы работы с визуальной частью информационной системы, то в другой возможно зрительно скрытой части этой системы методы работы с интерфейсом должны быть подобными, проще говоря, не отличаться и быть ожидаемыми. Также работа с графической частью информационной системы через интерфейс пользователя должна согласоваться с минимальными установленными, привычным для всех участников нормам;

Не избыточность. Это понятие означает, что заинтересованный сотрудник должен по возможности вводить только минимальную информацию необходимою для работы или управления системой;

Непосредственный доступ к подключённой системе помощи. В процессе работы нужно предусмотреть, чтобы наша информационная система снабжала среднестатистического пользователя жизненно необходимыми директивами работы с графической системой. Система помощи, разумеется, должна отвечает

трем основным факторам: качество и естественно количество гарантируемых команд; характер всплывающих уведомлений об ошибках и подтверждения того, что система делает. Сообщения об ошибках жизненно необходимо должны быть полезны и главное понятны простому пользователю;

Гибкость. Насколько хорошо пользовательский интерфейс нашей информационной системы может обслуживать ответственных специалистов с различными знаниями персонального компьютера? Для совсем неопытных специалистов интерфейс пользователя может быть сконструирован как ветвящаяся иерархическая структура строгого подчинения, а для более опытных пользователей как сухие команды, стандартными сочетаниями клавиш обычной клавиатуры и вводимые параметры. Необходимо учитывать привычное для пользователя место размещения искомой информации на главном экране. Количество выводимой за раз информации, количество отображаемой на экране информации, на профессиональном языке называется простым термином «экранной плотностью». Проведенные исследования этого явления показали, что, чем меньше «экранная плотность» сообщений, тем отображаемая на нем информация легче читается и по сути наиболее доступна и понятна для специалистов. Навигация в приложении должна обязательно соответствовать определенным требованиям [34]: принципы организации навигации по элементам формы должны быть едиными для всех составных частей системы благодаря этому свойству обычный пользователь может легко и быстро ориентироваться и выбирать нужные разделы, находясь на любой странице системы; на каждой странице системы должно присутствовать наименование текущего раздела системы в котором мы сейчас находимся; необходимо каким-либо способом обозначать в меню тот уровень или раздел, на котором пользователь находится в данный момент; ссылки в виде кнопок на любой странице системы должны выглядеть стандартно и давать пользователю однозначный сигнал о своей функциональности; вывод предупреждений перед совершением критического действия (удаление данных); вывод предварительных результатов в разделах посвященным программным мероприятиям; запоминаемость, насколько хорошо пользователь

будет ориентироваться в системе спустя некоторое время не взаимодействия с ней; минимизация ошибочных действий пользователя, исправление опечаток, замена запятой на точку, подстановка десятичных нолей.

2.3.1 Методы проектирования графического интерфейса

На этом этапе осуществляется анализ требований, предъявляемых к разрабатываемым компонентам, формализуется функциональность и определяются объективные критерии. На этом этапе закладываются основные концепции системы, влияющие абсолютно на все показатели качества её интерфейса.

Для того чтобы корректно и полно поставить задачу разработки, необходимо выполнить следующее [34]:

- формализацию контекста использования;
- формализацию объективных критериев успеха;
- определение необходимой функциональности системы;
- анализ целей;
- анализ действий пользователей;
- определение низкоуровневых и высокоуровневых функций;
- формализацию бизнес-ролей пользователей;
- формализацию функциональности;
- формализацию сценариев действий пользователей;
- обзор интерфейса конкурирующих систем; формализацию привычек и ожиданий пользователей.

2.3.1.1 Формализация объективных свойств системы

На данном этапе разработки интерфейса пользователя выделяются актуальные на данный момент или объективные критерии оценки которые на прямую влияют на эргономичность интерфейса пользователя. От этого зависит на прямую эффективность выполнения поставленной задачи. В результате это выливается в удовлетворенность пользователей. Собирается вся имеющаяся в наличии информация по данному вопросу и создается самое, что не наесть реальное техническое задание на разработку и проектирование графического интерфейса, например: Одна и та же задача, как правило, повторяется периодиче-

ски с постоянной частотой, при том, что группа наших пользователей в данный момент времени довольно большая. В связи с этим нам необходимо сосредоточить и акцентировать их внимание на возможной высокой эффективности использования предлагаемого варианта. Что позволит значительно снизить количество, как правило, одних и тех же повторяющихся человеческих ошибок. Грубо говоря, на входе доступ к необходимой документации, а на выходе получаем список объективных формализованных условий достижения успеха в нашем случае формы вывода информации с показателями эффективности.

2.3.1.2 Обоснование желательной минимальной многофункциональности информационной структуры

На первом цикле нам желательно вычислить минимальную функциональность нашей будущей информационной структуры. Это невероятно важный этап по сравнению с другими этапами, поскольку именно здесь в этом этапе непосредственно лаконичность будет характеризовать весь графический интерфейс пользователя. На этом этапе очень важно понять и проанализировать, что необходимо оставить и фактически немыслимо убрать из уже действующей схемы какие-либо подсистемы. Во-первых, все программы по сей день создаются по функциям. Трудящиеся пользователи не хотят переучиваться для применения совершенно новых функций системы взамен старых. Это означает только то, что по нашему мнению ненужная функция подсистемы будет постоянно мигрировать из версии в версию, наращивая своим присутствием размеры нашей программы, снижая её устойчивость программного обеспечения и в целом быстродействие всей системы, при этом засоряя собой и бес того нагруженный визуальный интерфейс пользователя (что повлияет на продолжительность разработки которая обязательно увеличивается).

Как повелось, предписания к функциональности информационной системы всегда исходят от заказчика. Такие требования разбивают на два типа: 1 претензии исходят имеющихся пользователей 2 другие информационные системы, на которые ориентируют пользователи. К нашему глубочайшему разочарованию, оба этих источника очень недостоверны. Как правило, в итоге мо-

жет оказаться, что сильное желание пользователя получить какую-то новую функцию в программном обеспечении, порождено не действительной их необходимостью в ней, а конкретно не правильной компоновкой интерфейса, что привело к проблемам нашей системы.

Программное обеспечение же, имеющиеся на рынке, страдают теми же проблемами. Это безусловно означает, что принимать в внимание требования к сторонним системам, конечно же, следует, нос осторожностью. Существуют только два зарекомендовавших себя другими словами основных фундаментальных метода для определения необходимой минимальной рациональности системы, мониторинг всех задач и анализ всевозможных действий наших пользователей. Все эти не тривиальные способы формально не конфликтуют друг с другом, более того, в процессе обозначения функциональности обязательно задействовать оба.

2.3.1.3 Анализ поставленных задач

Программисту в современном мире необходимо четко понимать, что среднестатистическим пользователям не нужны навороченные инструменты сами по себе, им нужны только результаты их работы полученные без усилий. Сейчас мало кому нужен текстовый процессор сейчас нужна программа позволяющая с удобством писать большие тексты. Это означает только то, что сами по себе разнообразные функции не нужны. Всем пользователям жизненно необходимо средство вообще, с помощью которого можно выполнять какую-либо работу без усилия.

Ни в коем случае не стоит дать себя обмануть никому ненужной излишней конкретикой, т.е. описанием того, какова должна быть будущая палитра инструментов. В большинстве случаев, одного и того же результата можно добиться несколькими совершенно разными способами, при этом очень важно не только реализовать уникальный отличный от других способ, но и выбрать самый лучший. Анализ целей ответственного исполнителя как раз и позволяет избежать ненужной проволочки с конкретикой.

Результатом этого этих шагов должен являться конкретный список поставленных целей. Например:

- внесение данных простое и удобное;
- вывод промежуточных результатов в главном окне;
- вывод оперативных отчётов одним нажатием.

После того как истинные конкретные цели наших пользователей локализованы, приходит то самое время выбирать единственно верный способ разработки функции, для этого должен использоваться второй метод.

2.3.1.4 Анализ взаимодействий будущих пользователей

Осуществление почти всех поставленных целей требует от всех без исключения пользователей совершения соответствующих действий. В очень сложных компьютерных информационных системах сами по себе выбранные стратегии пользовательских действий влияют на предписания к функциональности. В компьютерных же информационных системах взаимодействие обычно намного сложнее и запутаннее, при этом формальный логический анализ зачастую неприемлем. Единственным выходом из сложившейся ситуации является постоянное слежение за людьми, выполняющими поставленную миссию в рамках компетенции, пользуясь уже предоставленными им инструментами, а именно информационными системами наших конкурентов. Очень часто хорошим источником полезного материала для мониторинга часто служит мониторинг итогов работы профессионалов, если оказывается, что результат работы фактически не зависит от используемого механизма, это значит, что необходима только та гармоничность, которая оказала воздействие на итог (т. е. подсистемы, которыми никто не попользовался, не нужны).

2.3.1.5 Простые и высокоуровневые клиентские функции

На сегодняшний день выделяют всего два концептуально не похожих друг на друга механизма действий к упрощенному вычислению необходимой минимальной функциональности информационной системы.

Первый подход. Информационная система должна быть оснащена максимально возможным количеством разнообразных функций, однако при таком

подходе как показывает практика итоги подавляющего большинства являются результирующим обобщением совершенно других пользовательских функций.

Второй подход. Современный разработчик снабжает свою систему огромным комплектом основных условно элементарных функций проектируемой системы, из которых среднестатистический пользователь может собрать наиболее сложные парой очень необходимые ему функции.

Эти два описанных совершенно разных подхода абсолютно естественно имеют свои незначительные недостатки, однако есть и неоспоримые достоинства при применении этих подходов. Концептуальный метод, это метод когда общее количество действующих пользовательских функций сильно лимитировано, как следствие это позволяет очень сильно упростить пользовательский софт, однако при использовании этого подхода от нашего пользователя однозначно требуется очень глубокое понимание, как системы в целом, так и то, как из многих простеньких подсистем собирать подсистемы более сложного порядка. Технология, при которой предполагается не только упрощенные функции и сложнейшие высокоуровневые конструкции, что позволяет, прибегая к не большим усилиям обеспечивать, безусловно, наиболее большую производительность труда, однако от пользователя потребуются более глубокие знания о том, как, а чаще будет возникать вопрос, где отыскать эти самые высокоуровневые подсистемы и как с этими системами теперь работать, при том, что они обременяют функционал. Кроме того, ещё остается возможность компромисса: всегда есть перспектива включить в нашу систему дополнительные средства автоматизации процессов, чтобы наши пользователи получили возможность проектировать и создавать свои пользовательские функции. Этот изящный подход на данный момент времени является востребованным.

2.3.1.6 Формализация функциональности системы

Опираясь на полученную ценную информацию, выработанной на вышеописанных этапах, можно приступить к окончательному формированию списка функциональных способностей совершенно новой версии схемы. Переформированное на прошлых этапах техническое задание зачастую не содержит неко-

торой части необходимой пользователю функциональности либо вмещает функциональность, которая не требуется пользователям в настоящий момент времени. На входе доступ к пользователям. На выходе мы ожидаем получить минимальное описание функциональности проектируемой информационной системы.

2.3.1.7 Конструирование расположения пользовательских экранов системы

Руководствуясь на сценариях выполняемой пользователями работы и ролях распределённых между пользователями, по немного формируется новая структура пользовательских экранов информационной подсистемы, то есть совершенно логичным образом определяется минимально необходимое пользователю число экранов, информативность каждого из них, навигационные коммуникации между экранными формами, формируется взаимосвязанная подсистема меню и других навигационных компонентов. По сути, на этом витке формируются и различаются отдельные структурные блоки системы, и определяется механизм, как именно эти блоки осуществляют взаимодействие между собой. Под структурными блоками будем подразумевать группировку функций, связанных по переназначению. Проектирование графического интерфейса пользователя общей схемы состоит из двух параллельно протекающих процессов: формализация независимых блоков системы и определение взаимосвязей между ними.

2.3.1.8 Конструирование системы навигации пользователя На основе прежде выработанной нами в предыдущих блоках формализованной структуры пользовательских экранов на стадии проектирования определяется весьма приемлемая схема обхода экранных форм пользователем и проектируется с формализацией элементов её графический детально проработанный интерфейс пользователя. Схема навигации пользовательского графического интерфейса наглядно показывает порядок распределения пользовательских функций между экранными формами нашей стратегии. Наша навигационная схема определяет, как наши пользователи будут перемещаться между сформулированными зада-

чами. Как правило, настоящий момент времени, на этом технологическом этапе нет необходимости в создании отдельного отчета; разработанный графический интерфейс пользователя в будущем излагается в докладе, посвящённом низкоуровневому проектированию интерфейса пользователя. На входе структура разрозненных графических экранов. На выходе описание навигации по нашей системе.

2.3.1.9 Низкоуровневое конструирование пользовательского интерфейса

На данном этапе осуществляется конструирование клиентского интерфейса и экранных форм системы. Конструирование главных экранных форм пользователя. На данном шаге разрабатываются интерфейсы пользователя нашей информационной системы. Которые подвергаются тестированию. Опираясь на обобщенные материалы сделанных нами выводов из формализации условий для достижения успеха в построении интерфейса и порядка взаимодействий пользователей системы вырабатывают новые тестовые задания для наших пользователей, которые по нашему замыслу выполняются всеми заинтересованными пользователями с дальнейшей фиксацией результатов всех важных особенностей активности. После того как это будет сделано переходим к этапу на котором выполняется подсчет всех соответствующих нашей системе параметров и сопоставляем их с заданными параметрами ранее. Учитывая полученные результаты выполнения задания пользователями, графический интерфейс пользователя либо корректируется, либо остаётся в неизменном виде.

Конструирование второстепенных пользовательских экранных форм. На данном этапе разрабатываются графические интерфейсы второстепенных пользовательских экранов нашей информационной системы. К ним относятся диалоговые окна и всевозможные всплывающие сообщения системы. Итоговое всеобъемлющее тестирование экранных форм пользователя осуществляется по той же схеме. На основании априорных целей предполагаемого успеха пользовательского интерфейса и вариантов взаимодействий пользователей информационной системы вырабатывают заранее подготовленные тестовые задания, которые снова выполняются всеми нашими пользователями, снова проводится

фиксацией всех ошибочных и без ошибочных действий. После этого выполняется подсчет достигнутых показателей и сравнение их с заданными. Исходя из теоретических материалов, было принято решение о вводе в структуру пользовательского интерфейса следующих элементов:

- вывод вспомогательной информации в интерфейсе, по нажатию показываем элементы управления рисунок 7;
- изменение интерфейса в зависимости от параметров муниципальной программы рисунок 8.

Эти манипуляции с интерфейсом пользователя необходимы, так как вместить всю необходимую информацию на переднем плане не удастся. В силу большого объёма аналитических показателей.

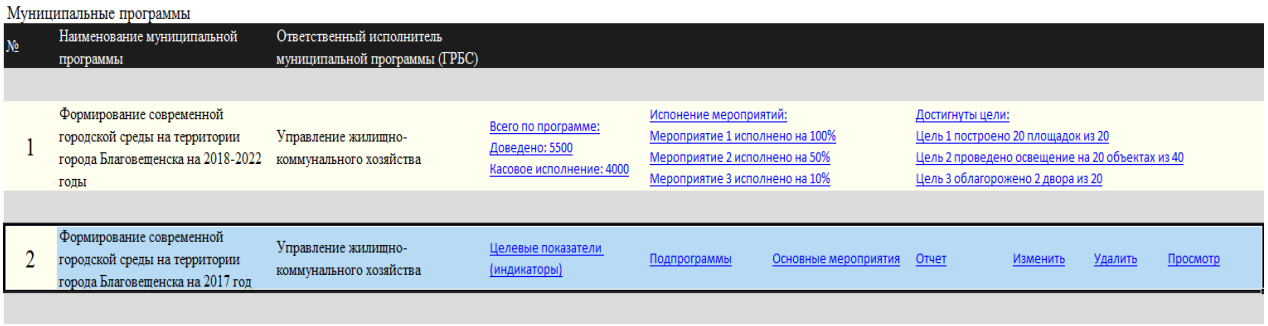


Рисунок 7 – Вывод вспомогательной информации

Рисунок 8 – Изменение вида интерфейса

Ожидаемый результат от применения манипуляций с интерфейсом:

- простота его изучения или овладения графического софта пользователя информационной системы и в итоге устойчивого запоминания системы;

- максимально точно определяется время устойчивого овладения графической части, и длительность по времени хранения полученных знаний в памяти;
- скорость и простота достижения непосредственных результатов поставленных задач при применении софта системы;
- удовлетворенность пользователей при эксплуатации выработанной системы.

Для улучшения визуального оформления использовался язык разметки CSS который позволяет построить web-интерфейс ни чем не отличающийся от интерфейса толстого клиента или обычного приложения windows [20].

2.4 Проектирование архитектуры программного обеспечения

Сравнение клиент-серверного приложения Windows forms и WEB приложения. Наиболее значимые параметры, выбранные для сравнения: производительность системы, надежность, ширина потока сетевого трафика, уровень загрузки администратора системы, обеспечение технической безопасности информационной системы и распределение нагрузки все это критерии оценки клиент-серверной схемы реализации приложения. Эти параметры тесно взаимосвязаны и влияют на производительность системы [32].

Выбор осложняется тем, что невероятно трудно дать точное определение двум рассматриваемым архитектурным подходам. В классической уже зарекомендовавшей себя схеме клиент - сервер, часть выполняемая на стороне клиентского автоматизированного рабочего места представлена классическим (как правило Windows) приложением, которое принято именовать «толстый клиент», проще говоря типовое windows приложение, разработанное с помощью проприетарного пакета Microsoft Visual Basic или C #. Такое название эта схема получила в связи с тем, что помимо уровня представления информации содержит еще и бизнес-логику прикладной задачи прям на клиентском автоматизированном рабочем месте. Такой клиент как правило работает не только с реляционной базой данных, но и с файл-сервером. Типовое браузерное Интернет-приложение именуют «тонкий клиент», в подавляющем большинстве слу-

чаев оно, используется лишь для представления данных в виде графики. Вся бизнес-логика все расчёты в этом случае выполняется на удаленном сервере приложений, который и обращается к системе управления базами данных через специальный драйвер. Однако на сегодняшний день существуют и другие реализации архитектуры клиент-сервер, где часть бизнес - процессов перенесена на специальные выделенные компьютеры. Такие информационные системы уже ближе к классической Web-архитектуре. Или, наоборот, могут быть созданы Web-приложения, в которых часть бизнес-логики выполняется на основе кода на PHP или мини-приложения Java. Выбор пал на клиент серверную систему с WEB интерфейсом [2].

Подготовку технического задания на разработку ПО провело управление экономического развития и инвестиций совместно со мной и ответственными исполнителями. В техническое задание вошли формы отчётов в соответствии с нормативными актами и пожеланиями ответственных исполнителей по поводу интерфейса будущего ПО. Проектирование ПО включило в себя изучение методов и принципов создания визуальных моделей взаимодействия бизнес процессов, визуализации структуры базы данных, модулей программы и взаимодействия классов ПО. Визуализация позволила получить ясную картину взаимодействия модулей, связи экранных форм и будущий интерфейс пользователя. После чего в техническое задание были внесены правки. Этап разработки архитектуры включает в себя выбор аппаратных и программных средств. В нашем случае выбор был сделан в сторону клиент-серверного ПО с WEB интерфейсом рисунок 9. После этого можем переходить к проектированию ПО модульно, то есть какие модули будут реализованы в этом ПО. Для целей проектирования и визуализации взаимодействия модулей используется UML диаграмма User Case (Диаграмма прецедентов) представлена приложении А. Выполнена она с помощью условно бесплатного интернет ресурса Creately [4, 37].

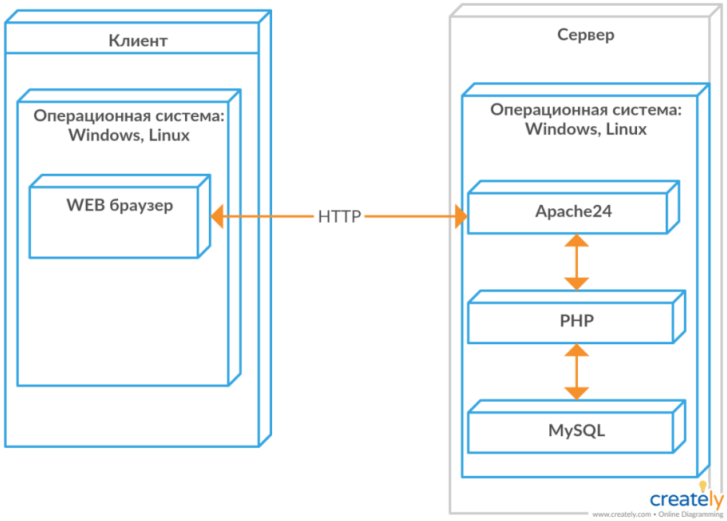


Рисунок 9 – Архитектура программного обеспечения

Выбор базы данных для бедующего ПО проходил по следующим критериям:

- простота в использовании; обширный функционал; безопасность; масштабируемость;
- цена;
- скорость обработки.

Исходя из этого, был выбран условно бесплатный сервер базы данных MySQL.

Основными функциями современной базы данных (банков больших данных) являются:

- долгосрочное хранение переданной информации и её защита;
- изменение хранимых данных;
- поиск информации в таблицах и отбор данных искомых по входящим запросам авторизованных пользователей;
- обработка больших данных и вывод полученных результатов.

База больших данных обязательно обеспечивает постоянное хранение информации и представляет собой сложную структуру поименованных совокупностей связанных по определенному признаку данных, организованных по определенным правилам, включающие общие принципы описания таблиц, хранения и сортировку данных.

Система управления современными базами данных является ни чем иным как пакет встроенных прикладных программ, предназначенных для создания, сопровождения и использования базы данных информационной системы. В нашем случае в качестве редактора базы данных было использовано бесплатное ПО HeidiSQL.

Прикладное программное обеспечение (приложение) в составе банка данных служит для обработки, как правило, больших данных, постоянных вычислений и формирования выходных представлений по заданной форме.

Логическая диаграмма таблиц базы данных проведена в приложении Г.

С программным обеспечением может взаимодействовать несколько категорий пользователей. К ним относятся:

1) ответственные исполнители или пользователи, осуществляет непосредственное внесение данных о муниципальной программе;

2) администраторы осуществляют заполнение справочников, создание учётных записей пользователей, разграничение прав пользователей.

Диаграммы взаимодействия пользователя и администратора представлены на рисунках 10 и 11 в виде диаграмм вариантов использования [5].

Рисунок 10 – Диаграмма взаимодействия администратора с программным обеспечением

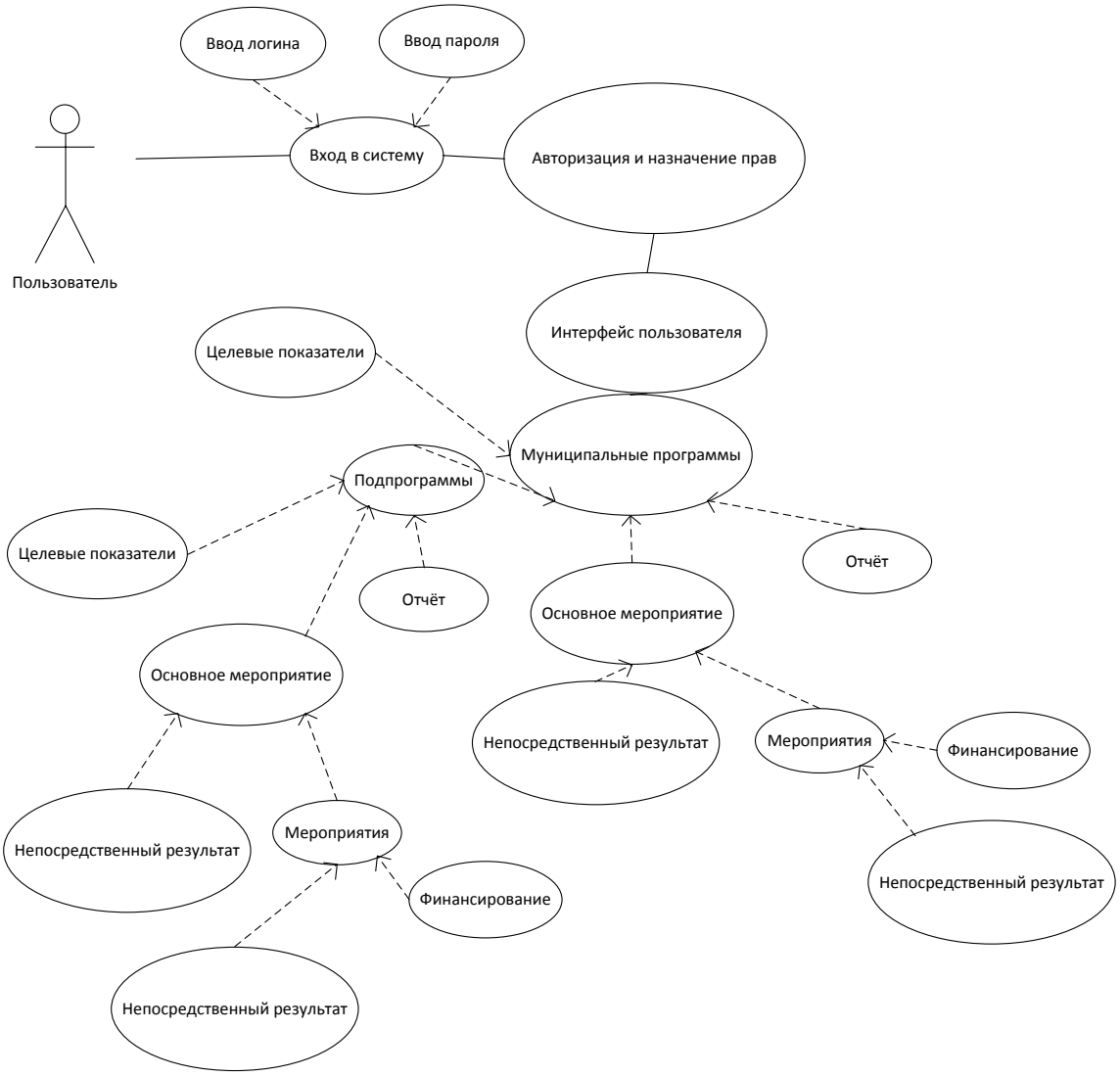


Рисунок 11 – Диаграмма взаимодействия пользователя с программным обеспечением

Программное обеспечение доступно постоянно. После удачного входа пользователя в программу у него появляется возможность вносить необходимые данные для пользователя и управлять учётными записями пользователей у администратора. Функции администратора включают в себя помимо управления учетными записями пользователей, назначения прав доступа ещё и заполнение системных справочников. Работа пользователя (ответственного исполнителя) наиболее разнообразна. Помимо самостоятельных элементов не требующих обязательного заполнения существуют элементы, без заполнения которых продолжение ввода данных невозможно. Это связано с тем, что в системе муниципальных программ строго подчиненные этапы реализации программы, не предполагающие отступлений [26].

На рисунке 12 представлена диаграмма деятельности в стандарте UML, на ней максимально детально изображены последовательные действия пользователей системы, состояния которых наглядно представлено на нашей диаграмме состояний в виде модулей. Под деятельностью (англ. activity) обычно понимается техническая спецификация действующего поведения в виде жестко определенного, строго последовательного а порой и многопоточного выполнения подчинённых (дочерних) элементов как правило вложенных друг в друга порой разных видов деятельности (activity) и отдельных запрограммированных действий англ. action, неразрывно соединённых между собой направленными потоками информации, которые идут из выходов одного узла или нескольких ко входам других узлов или другого узла [6].

Диаграммы деятельности в современном проектировании очень часто используются при моделировании существующих бизнес-процессов, порой технологических процессов, зачастую последовательных и даже параллельных модулей вычислений. Такие диаграммы деятельности состоят из очень ограниченного набора определенных (стандартизированных) фигур, соединённых вполне обычными на вид стрелками стандартного набора. Основные фигуры[22]:

Прямоугольники с закруглениями подробное описание – действия оператора в системе;

Ромбы – принятие какого либо решения;

Широкие полосы – начало (разветвление) и окончание (схождение) ветвления действий;

Чёрный круг – показывает начало процесса (начальный узел);

Чёрный круг с обводкой – означает окончание процесса (финальный узел).

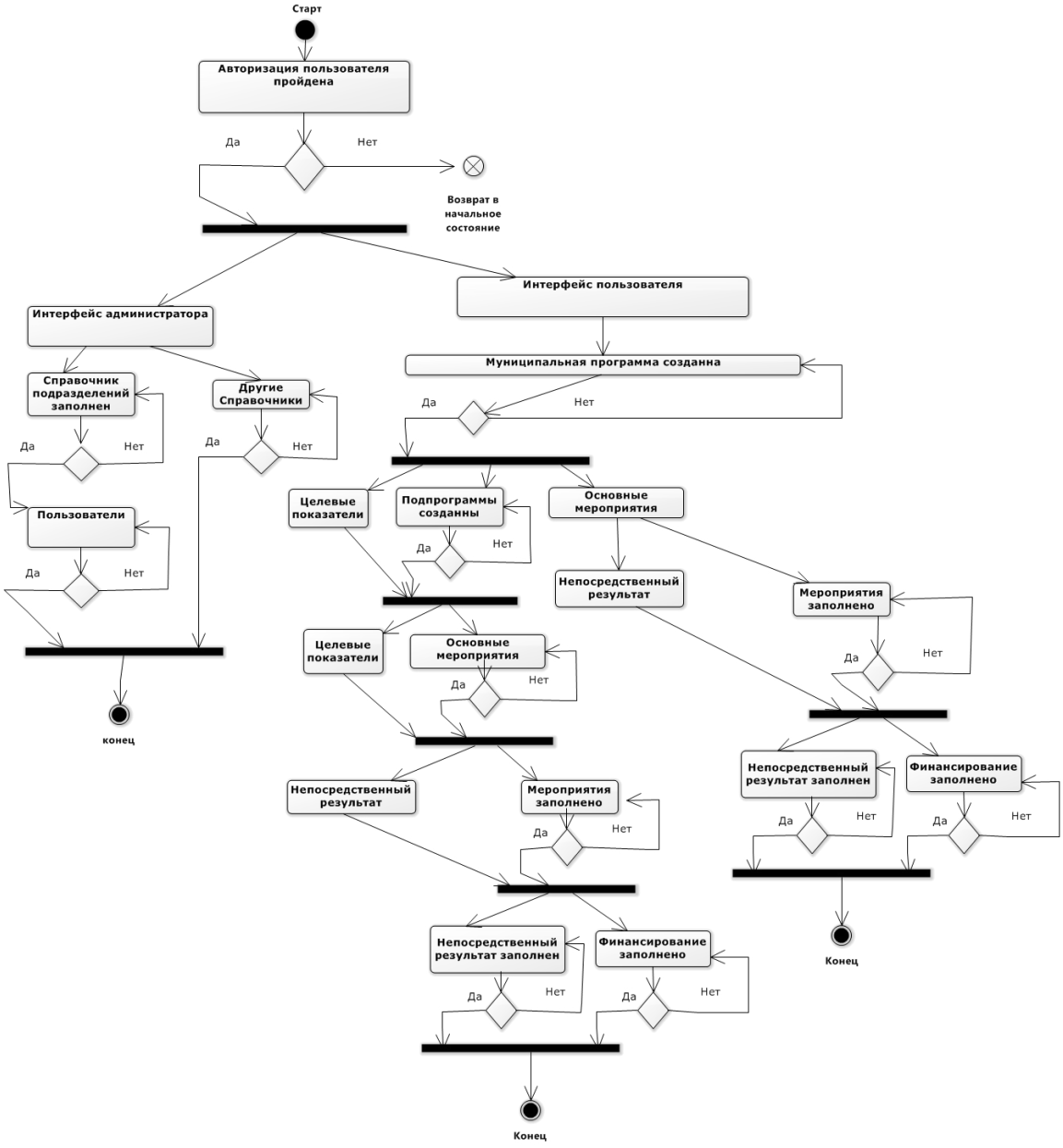


Рисунок 12 – Диаграмма деятельности

Перед внесением данных исполнитель выполняет вход в программу. После этого появляется возможность вносить необходимые данные. При этом пользователь имеет возможность выбрать этап создать новую программу или продолжить работу с ранее созданной. На представленной ниже диаграмме изображен процесс от создания новой муниципальной программы до вывода отчёта по ней. Во время работы пользователь постоянно получает на экран монитора отображение интерфейса и всплывающие сообщения системы при допущении ошибок. Диаграмма последовательности приведена на рисунке 13.

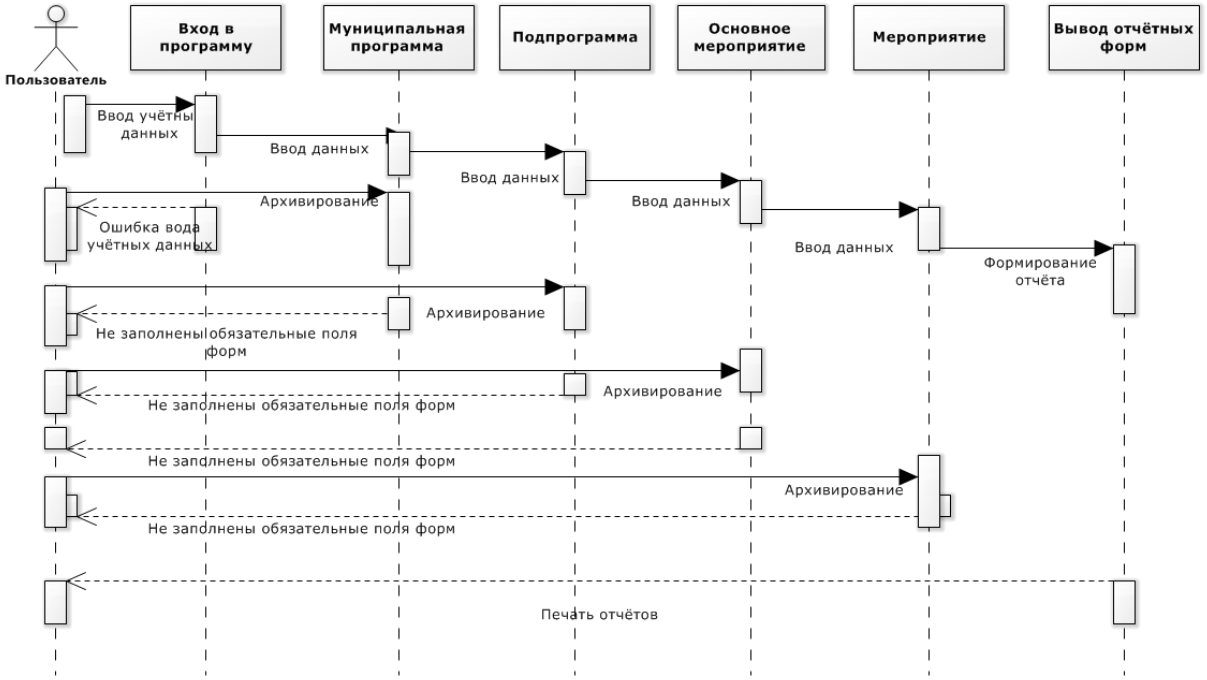


Рисунок 13 – Диаграмма последовательности

В рабочем цикле программы присутствуют циклические операции это – архивирование данных которое происходит постоянно, и корректировка данных без которой просто не обойтись. Корректировка возможна в любой момент. На диаграмме (рисунок 14) она изображена после вывода отчёта по причине того, что большинство ошибок легко обнаружить именно в отчёте.

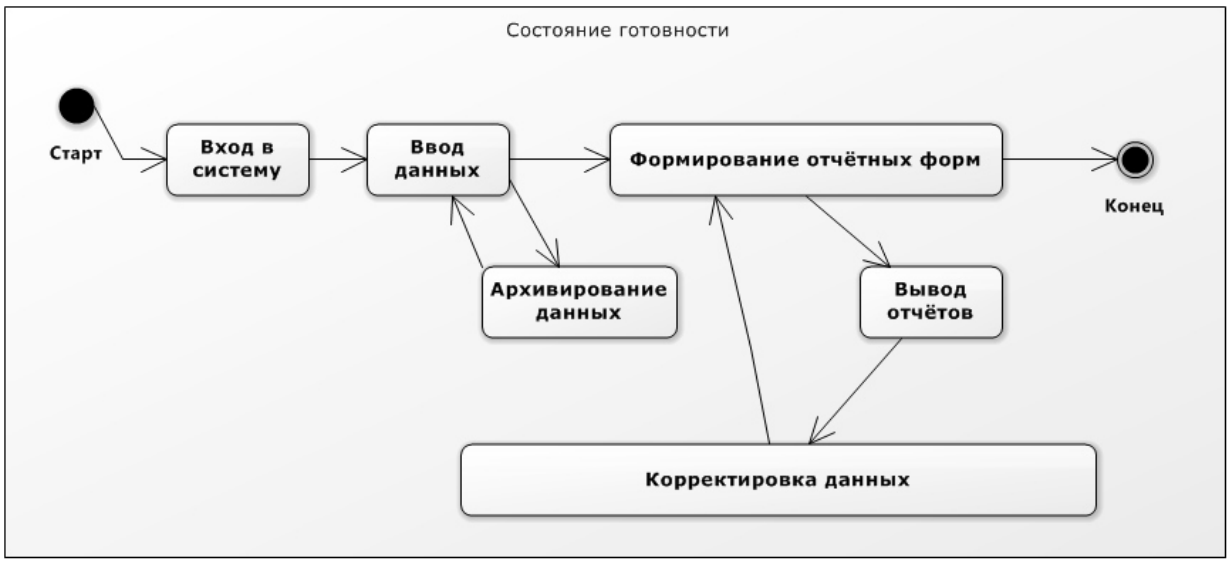


Рисунок 14 – Диаграмма состояний

Программный обеспечение делится на несколько компонентов. В него входят:

1) Модуль авторизации, связанный с базой данных через модуль работы с базой данных;

2) Графический интерфейс администратора;

3) Графический интерфейс пользователя;

4) Модуль доступа к базе данных с операциями чтения и записи данных;

5) Справочная система содержащая руководство пользователя Manual.chm.

6) Модуль подготовки отчётов.

Модули представлены в диаграмме компонентов (рисунок 15).

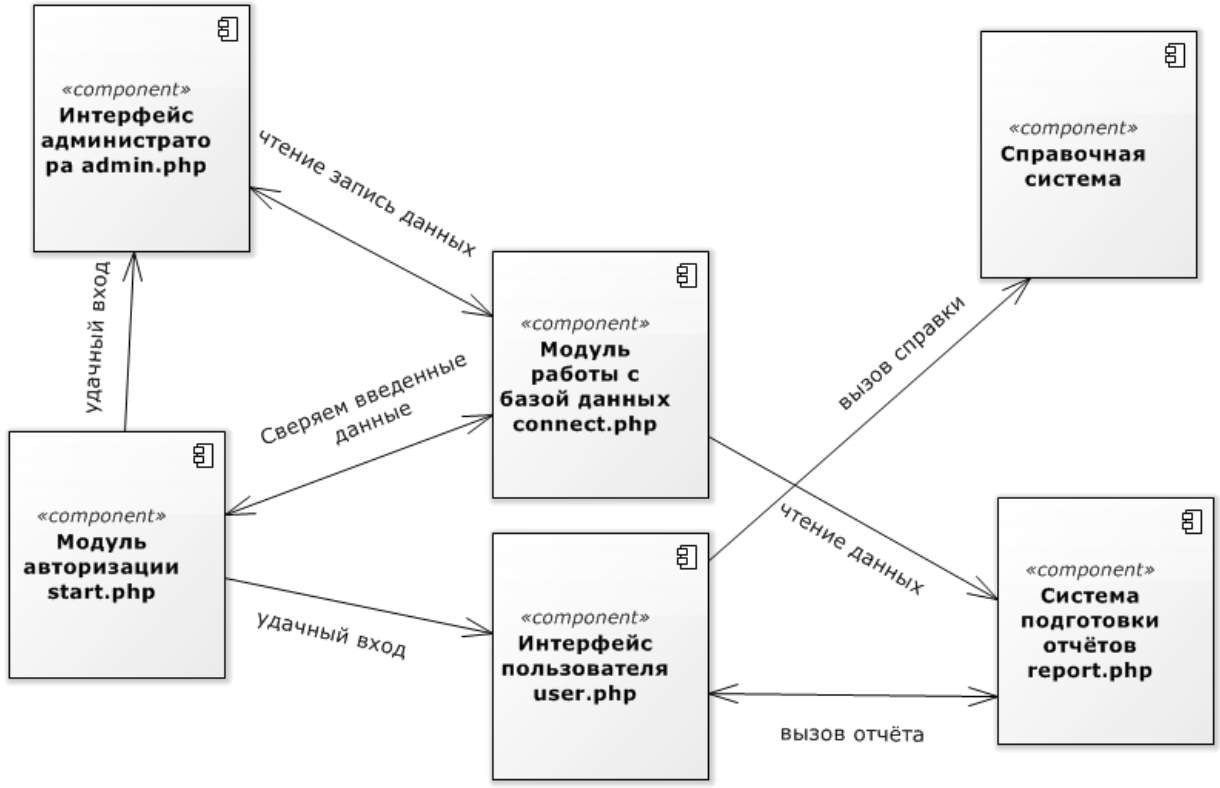


Рисунок 15 – Диаграмма компонентов

Выбор спиральной модели жизненного цикла проектируемого нами нового программного обеспечения (рисунок 16) обусловлен тем, что детальная формализация модулей программного обеспечения могла меняться по желанию заказчика [9]. По сути функции программных модулей могут постоянно расши-

ряться, а иногда некоторые элементы исключаться из проекта. Часто происходила ситуация по переносу (группировке) экранных форм по логическому признаку в связи с изменениями логики и структуры программного мероприятия.

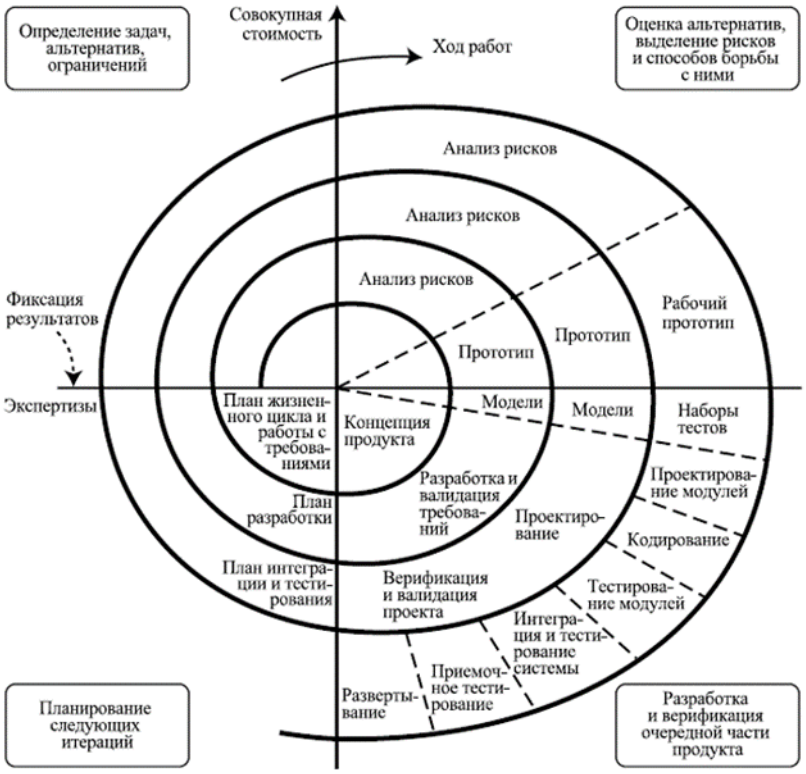


Рисунок 16 – Спиральная модель жизненного цикла программного обеспечения

На протяжении всего жизненного цикла проектируемого нами программного обеспечения будут, осуществляется дополнения как минимум отчётных форм, как максимум дополнение модулем автоматического формирования доклада руководителю о состоянии реализации муниципальных программ за определённый промежуток времени в который будут включены графики и диаграммы, наглядно представляющие достигнутые результаты. Таким образом, данная модель жиз. цикла прекрасно подходит для решения поставленной задачи.

2.5 Проектирование модулей программного обеспечения в виде диаграммы прецедентов

Для целей проектирования и визуализации взаимодействия модулей информационной системы я использовал нотацию UML диаграммы User Case

(Диаграмма прецедентов). В нашем случае была выполнена с помощью интернет ресурса Creately [12, 27]. Основными категориями зарегистрированных в нашей информационной системе пользователей являются «Администратор» в лице ответственных сотрудников управления экономического развития и «Пользователи» в лице ответственных исполнителей от управлений администрации. Группа администраторы выполняют следующие задачи:

- регистрация пользователей; разграничение прав доступа;
- заполнение справочников;
- загрузка данных из информационной системы финансового управления; формирование доклада мэру.

Группа пользователей выполняет следующие задачи:

- внесение наименования мероприятий, целевых показателей, подпрограмм, основных мероприятий и мероприятий;
- внесение плановых и фактических значений целевых показателей, непосредственных результатов;
- внесение данных о финансировании и ходе освоения средств;
- формирования отчётности по капитальным и прочим вложениям в соответствии с нормативными правовыми актами (автоматизировано).

Диаграмма прецедентов визуализирует модульную структуру и связи между модулями. Благодаря этой диаграмме наглядно видно излишние процедуры и задавленности модулей (при их наличии). В результате её использования можно сократить количество модулей и, следовательно, затраты на разработку программного обеспечения. Диаграмма прецедентов представлена в приложении А в соответствии стандарту UML.

2.6 Проектирование и визуализация структуры базы данных

Базу данных будущего ПО было решено реализовать на сервере баз данных MySQL. Диаграмма базы данных представлена в приложении Б. Таблицы базы включают, идентификаторы всех сущностей. Причём каждая сущность имеет исчерпывающий набор идентификаторов так называемых родителей. Не-

смотря на кажущуюся избыточность и нарушение принципа реляционных баз данных такая структура в дальнейшем сильно сказывается на быстродействии системы [25, 29]. А именно в момент формирования отчёта по всем действующим программам. На пример если нам нужно сделать выборку финансирования мероприятий, то идём циклом по каждому мероприятию соответственно делаем запрос в базу со связанными таблицами. В то время как основное мероприятие и тем более муниципальная программа и подпрограмма остаются неизменными, а в запросе придётся делать проверку на наличие основного мероприятия, подпрограммы и программы. Такие запросы нагружают PHP сервер в связке MySql. На мой взгляд, проще и менее ресурсозатратно в момент старта отчёта набираем массивы всех сущностей и работаем уже с массивами набранных данных. Эта техника была проверена мною на практике, время обработки существенно отличается в пользу массивов данных. Несмотря на то, что один запрос выполненный в клиенте базы данных занимает время, исчисляемое миллисекундами построение отчёта при выборке большого количества данных, занимает значительное время. За которое пользователь системы начинает беспокоиться и выражать своё не довольствие.

2.7 Проектирование модулей программного обеспечения

Перед этапом разработки неплохо определиться, что собственно будет представлять собою ПО, какие составляющие войдут в него, как будут влиять на бизнес процесс, кто исполнитель и его роль и, что будет на выходе. Для этих целей была использована нотация IDFE0 в построении диаграмм. Диаграмма композиции представлена в приложении В. Которая представляет собой обобщенную схему программного обеспечения в которой отображены управляющие процессы, такие как нормативные документы. Входящие данные [21]:

- информация о муниципальных программах, подпрограммах и т.д.;
- заполнение справочников системы;
- данные бюджета;
- исполнение мероприятий, достижение результатов и индикаторов.

В роли механизма выступают управление экономического развития и ответственные исполнители. Управление экономического развития выступает в двух ролях как администратор и как ответственный исполнитель по муниципальным программам. В результате всего этого должно получается на выходе:

- доклад мэру;

- аналитические отчёты о выполнении мероприятий доступные в любой момент времени;

- визуализированные данные для своевременного принятия управленческого решения.

Диаграмма декомпозиция нотации IDFE0, представлена в приложении Г. Диаграмма показывает модули, из которых будет состоять наше программное обеспечение. Модуль «Администрирование» служит для разграничения ролей пользователей, наполнения справочников единиц измерения, справочника ответственных подведомственных учреждений, импорта данных об исполнении бюджета из информационной системы финансового управления, регистрацию пользователей.

Модуль «Обработка данных» реализует собственно сбор данных о программах, подпрограммах, основных мероприятиях, мероприятиях, финансировании, целевых показателях и непосредственных результатах плановых и достигнутых значениях. Пользователь, обладающий достаточными правами доступа, имеет возможность создавать новые «ветви событий», вносить правки в сущности объекта. Права доступа имеют принцип наследования, то есть если пользователь родительской подпрограммы создает новую «ветку мероприятий», то он автоматически имеет полный доступ на всё, что создаст. Кроме этого он может добавлять исполнителей к мероприятиям, которые в свою очередь будут иметь права на создание редактирование финансирования. На выходе этот модуль даёт упорядоченные массивы данных для модуля «Подготовка отчётных форм».

Модуль «Подготовка отчётных форм» имеет задачу вывода визуализированных форм с информацией о ходе исполнения мероприятий, муниципальных

программ на основе введенных пользователями данных в модуль «Обработка данных». Отчётные формы, подготовленные модулем весьма разнообразны:

- доклад мэру;

- множество табличных отчётов в соответствии с НПА;

- диаграммы и графики необходимые для принятия управленческих решений.

- материалы (в виде графиков и диаграмм) для подготовки докладов руководителям.

Этот модуль доступен всем пользователям и не имеет разграничений по правам доступа. Но модуль «знает» кто в данный момент формирует отчёт. В соответствии с этими данными предлагает пользователю (расстановка чек боксов) условия выборки отчётов.

2.8 Проектирование взаимодействия классов программного обеспечения в нотации UML

Взаимодействие классов проектируемого программного обеспечения представлено в приложении Е. Наследование классов представлено двумя типами наследования: ассоциацией и композицией. Диаграмма наглядно показывает взаимодействие классов, какой класс, от какого наследуется и имеет поля. Какие поля обязательны к инициализации. Диаграмма классов показывает сложную структуру взаимодействия, которую при возможности следует упростить, однако, на мой взгляд, представленная диаграмма не может быть упрощена и представляет собой законченную структуру с минимальным набором полей или свойств. Основные классы, которые многократно вызываются – это [7]:

- ConnectDB класс подключения к базе данных и выполнения запросов;

- авторизация класс предоставления доступа к системе и при необходимости расширенных прав;

- программы класс, который является родителем для всех остальных классов;

- класс подготовки отчётов.

Таким образом, выделяем три основных класса без которых функционирование программного обеспечения как информационной системы невозможно.

Класс подготовки отчётов в этой схеме опущен, поскольку может использоваться без предоставления прав доступа, то есть он доступен всем и всегда, имеет свой собственный коннектор к базе данных. По сути, представляет собой самодостаточный модуль и может использоваться без других классов. С одной – это сделано для предоставления доступ к отчётности незарегистрированным пользователям (желание заказчика). С другой стороны, не имея параметров подключения к системе отчётности получить к ней доступ вряд ли удастся.

2.9 Выводы по главе

Разработка и внедрение программного обеспечения для осуществления оперативного мониторинга исполнения программных мероприятий администрации города Благовещенска является сложной и актуальной задачей. Для её реализации было сформулировано техническое задание, проведено изучение теоретических материалов, посвящённых методам и практической реализации проектирования ПО. В частности, было изучено:

- интернет ресурс Free Online Diagram Editor построения диаграмм нотации IDFE0;

- программное обеспечение построения логических и физических проектов баз данных MySQL Workbench;

- условно бесплатный интернет ресурс Creately построения диаграмм стандарта UML, с помощью его построена диаграмма прецедентов;

- ПО для построения диаграмм взаимодействия классов Dia.

Получены теоретические знания и практические навыки в части построения графического интерфейса пользователя, конкретно какие критерии определяют эргономичность интерфейса. Определены этапы поиска удобного интерфейса для пользователя. В целях тестирования идеи разработан макет части интерфейса пользователя. По результатам тестирования было принято решение о доработке и внедрении новой улучшенной версии интерфейса. В целях улучшения восприятия пользователем информации с экрана проведено цветовое

дифференцирование результатов для этого изучены основы CSS разметки и реализованы в виде стилей для блоков информации [14]. Несмотря на то, что разработка пользовательского интерфейса является серьёзной проблемой, так как каждый пользователь видит информационную систему по-своему. Найти в таких условиях интерфейс, который устроит всех очень сложно. Решение было найдено путём проведения анкетирования потенциальных пользователей.

Получены практические навыки проектирования программного обеспечения (информационной системы) с помощью выше описанных инструментов. Построена логическая структура базы данных будущего ПО, в которую внесены данные справочников и муниципальных программ.

Изучены основы построения реляционных баз данных. Развернут сервер базы данных MySQL, развернут интерпретатор скриптов PHP и WEB сервер Apache. Настроено взаимодействие модулей развернутого ПО. Разработаны основные классы информационной системы, проведено тестирование их взаимодействия. Написание скриптов осуществлялось в бесплатном редакторе Notepad++. Спроектированы все экранные формы необходимые пользователю для выполнения своих служебных обязанностей. Проектирование визуальных форм вывода информации и макетов форм отчётности осуществлялось в OpenOffice Calc с последующим переводом в HTML.

3 ОПИСАНИЕ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ И ВВОД В ОПЫТНУЮ ЭКСПЛУАТАЦИЮ

- Разработанное программное обеспечение имеет следующие функциональные возможности:
- формирование муниципальных программ и дополняющих документов;
- формирование структуры и содержания разделов муниципальной программы внесение изменений;
- формирование планов, планов-графиков реализации программных мероприятий вышестоящей по иерархии муниципальной программы и иных дополняющих документов;
- формирование отчётных данных о финансировании мероприятий муниципальной программы;
- формирование отчётных данных о ходе выполнения мероприятий муниципальной программы;
- формирование отчётных данных о достигнутых значениях целевых показателей (регламентированных индикаторов) муниципальной программы;
- формирование отчётных форм о ходе и результатах реализации муниципальной программы;
- расчёт оценки эффективности исполнения муниципальной программы согласно утверждённой методике;
- формирование отчётных форм о результатах оценки эффективности муниципальной программы;
- использование регламентированных индикаторов для наглядного отражения результатов проведенного анализа информации о ходе и результатах реализации запланированных муниципальных программ (соблюдения плановых сроков, достижения показателей);
- отображение информации в виде интерактивных элементов.

3.1 Описание базы данных

Разработанное программное обеспечение имеет разграничение по ролям доступа. Администратору системы доступно внесение информации в справочники системы. Более точно, это вид таблицы базы данных таблицы-справочники и таблицы-связки с множеством ключевых полей.

Информацию в таблицах можно разделить на два вида. На структурированные данные, которые описывают объекты, ключевые поля для связи между таблицами и полезные данные, которые описывают действия, процессы запрограммированные в процедурах, события которые происходят в структуре данных.

В справочниках содержатся сведения об объектах и субъектах, связях. В связках содержатся сведения о действиях, процессах, событиях.

В связках хранятся данные, взятые из таблиц справочников. Поскольку совершенно не рационально повторять одни и те же данные при описании объектов и при описании их взаимодействия, данные об объектах заносятся в так называемые справочники которые представляют собой такие же таблицы базы как и другие, а в таблицах-связках не хранятся данные объектов в чистом виде, а лишь ссылки на них. Таким образом, в связках хранятся данные по взаимодействию объектов (субъектов) и ссылки на сами объекты или внешний ключ. Эти «ссылки» являются первичными ключами в таблицах справочниках. Отличие справочника от ключевых полей связи выражается это в том, что все таблицы-справочники зачастую являются самостоятельными сущностями и не зависят от других таблиц, что означает, при чтении полезных данных некоторых таблиц-справочников можно в целом понять структуру хранения данных во всей базе, а таблицы-связки напротив практически никогда.

По классификации справочников мы имеем статический справочник, в котором данные об всех объектах, субъектах, связях никогда не подвергаются модификации.

Таблица 2 – Справочники ПО

Справочник	Название	Ссылки
Пользователи	users	dostup
Роли доступа	dostup	users programs pod_ programs osnovnoe_meropriyatie
		meropriyatie programs pod_ programs osnovnoe_meropriyatie meropriyatie
Единицы измерения	ed	Indicator Indicator_dost result_meropr result_plan result_result
Месяцы	mons	finans_meropr finans_osnmeropr result_meropr result_result
Периоды отчётов	period	Indicator_dost
Подразделения	podrazdelenie	otvetstvennie
Вид расходов	vid_rashodov	finans_meropr finans_osnmeropr
Ответственные	otvetstvennie	programs pod_ programs osnovnoe_meropriyatie meropriyatie meropr_no_prog

Таблица 3 – описание таблицы муниципальных программ programs

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	pod_ programs
number	CHAR	номер по документу	
name	VARCHAR	наименование	
year_nachaka	TEXT	дата начала	
year_okonchaniya	TEXT	дата окончания	
Indicator	VARCHAR	динамическое	
finans_fed	VARCHAR	динамическое	
finans_obl	VARCHAR	динамическое	
finans_mun	VARCHAR	динамическое	
finans_vne	VARCHAR	динамическое	
visible	INT	логическое удаление	
data_sozdaniya	DATE	дата создания	

Таблица 4 – описание таблицы муниципальных подпрограмм pod_programs

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	osnovnoe_meropriyatie
Id_prog	INT	ключевое поле	programs
name	VARCHAR	наименование	
number	CHAR	номер	
finans_fed	VARCHAR	динамическое	
finans_obl	VARCHAR	динамическое	
finans_mun	VARCHAR	динамическое	
finans_vne	VARCHAR	динамическое	
data_stsrt	CHAR	дата начала	
data_end	CHAR	дата окончания	

Продолжение таблицы 4			
visible	INT	логическое удаление	

Таблица 5 – описание таблицы муниципальных подпрограмм osnovnoe_meropriyatie

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	Ключевое поле	
Id_prog	INT	Ключевое поле	programs
Id_pod_prog	INT	Ключевое поле	pod_ programs
name	VARCHAR	наименование	
number	CHAR	номер	
finans_fed	VARCHAR	динамическое	
finans_obl	VARCHAR	динамическое	
finans_mun	VARCHAR	динамическое	
finans_vne	VARCHAR	динамическое	
data_stsrt	CHAR	дата начала	
data_end	CHAR	дата окончания	
visible	INT	логическое удаление	

Таблица 6 – описание таблицы муниципальных подпрограмм meropriyatie

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	
Id_prog	INT	ключевое поле	programs
Id_pod_prog	INT	ключевое поле	pod_ programs
Id_osn_meropr	INT	ключевое поле	osnovnoe_meropriyatie
name	VARCHAR	название	
name_fed_prg	VARCHAR	рег. название	

Продолжение таблицы 6			
name_reg_prg	VARCHAR	федеральное название	
data_stsrt	VARCHAR	дата начала	
data_end	VARCHAR	дата окончания	
kap	INT	признак капитального строения	
visible	INT	логическое удаление	

Таблица 7 – описание таблицы муниципальных подпрограмм finans_meropr

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	
Id_prog	INT	ключевое поле	programs
Id_pod_prog	INT	ключевое поле	pod_ programs
Id_osn_meropr	INT	ключевое поле	osnovnoe_meropriyatie
Id_meropr	INT	ключевое поле	meropriyatie
Id_otv	INT	ключевое поле	otvetstvennie
Id_mons	INT	ключевое поле	mons
mons	VARCHAR	месяц	
year	CHAR	год	
name	VARCHAR	название	
name_fed_prg	VARCHAR	региональное название	
name_reg_prg	VARCHAR	федеральное название	
data_stsrt	VARCHAR	дата начала	

Продолжение таблицы 7			
data_end	VARCHAR	дата окончания	
Kap_stroy	INT	признак кап. строения	
visible	INT	логическое удаление	
plan_fed	VARCHAR	план фед. бюдж.	
plan_obl	VARCHAR	обл.	
plan_gor	VARCHAR	гор.	
plan_vne	VARCHAR	внебюджет	
fackt_kass_fed	VARCHAR	кассовое исполнение	
fackt_kass_obl	VARCHAR	кассовое исполнение	
fackt_kass_gor	VARCHAR	кассовое исполнение	
fackt_kass_vne	VARCHAR	кассовое исполнение	
fackt_osv_fed	VARCHAR	освоение	
fackt_osv_obl	VARCHAR	освоение	
fackt_osv_gor	VARCHAR	освоение	
fackt_osv_vne	VARCHAR	освоение	
ostatok_fed	VARCHAR	остаток	
ostatok_obl	VARCHAR	остаток	
ostatok_plan_fed	VARCHAR	плановый остаток	
ostatok_fact_fed	VARCHAR	плановый остаток	
ostatok_kass_fed	VARCHAR	плановый ост.	

Продолжение таблицы 7			
ostatok_osvoen_fed	VARCHAR	плановый остаток	
ostatok_plan_obl	VARCHAR	плановый остаток	
ostatok_fact_obl	VARCHAR	плановый остаток	
ostatok_kass_obl	VARCHAR	плановый остаток	
ostatok_osvoen_obl	VARCHAR	плановый остаток	
ostatok_plan_gor	VARCHAR	плановый остаток	
ostatok_fact_gor	VARCHAR	плановый остаток	
ostatok_kass_gor	VARCHAR	плановый остаток	
ostatok_osvoen_gor	VARCHAR	плановый остаток	
prim	VARCHAR	примечание	

Таблица 8 – описание таблицы муниципальных подпрограмм finans_osnmeropr

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	
Id_prog	INT	ключевое поле	programs
Id_pod_prog	INT	ключевое поле	pod_ programs
Id_osn_meropr	INT	ключевое поле	osnovnoe_meropriyatie
Id_meropr	INT	ключевое поле	meropriyatie
Id_otv	INT	ключевое поле	otvetstvennie

Продолжение таблицы 8

Id_mons	INT	ключевое поле	mons
mons	VARCHAR		
year	CHAR		
name	VARCHAR		
name_fed_prg	VARCHAR	региональное название	
name_reg_prg	VARCHAR	федеральное название	
data_stsrt	VARCHAR		
data_end	VARCHAR		
Kap_stroy	INT	признак капитального строения	
visible	INT	логическое удаление	
plan_fed	VARCHAR	план фед. бюдж.	
plan_obl	VARCHAR	обл.	
plan_gor	VARCHAR	гор.	
plan_vne	VARCHAR	внебюджет	
fackt_kass_fed	VARCHAR	кассовое исполнение	
fackt_kass_obl	VARCHAR	кассовое исполнение	
fackt_kass_gor	VARCHAR	кассовое исполнение	
fackt_kass_vne	VARCHAR	кассовое исполнение	
fackt_osv_fed	VARCHAR	освоение	
fackt_osv_obl	VARCHAR	освоение	

Продолжение таблицы 8			
fackt_osv_gor	VARCHAR	освоение	
fackt_osv_vne	VARCHAR	освоение	
ostatok_fed	VARCHAR	остаток	
ostatok_obl	VARCHAR	остаток	
ostatok_plan_fed	VARCHAR	плановый остаток	
ostatok_fact_fed	VARCHAR	плановый остаток	
ostatok_kass_fed	VARCHAR	плановый остаток	
ostatok_osvoen_fed	VARCHAR	плановый остаток	
ostatok_plan_obl	VARCHAR	плановый остаток	
ostatok_fact_obl	VARCHAR	плановый остаток	
ostatok_kass_obl	VARCHAR	плановый остаток	
ostatok_osvoen_obl	VARCHAR	плановый остаток	
ostatok_plan_gor	VARCHAR	плановый остаток	
ostatok_fact_gor	VARCHAR	плановый остаток	
ostatok_kass_gor	VARCHAR	плановый остаток	
ostatok_osvoen_gor	VARCHAR	плановый остаток	
prim	VARCHAR	примечание	

Таблица 9 – описание таблицы муниципальных подпрограмм indicator

Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	
Id_prog	INT	ключевое поле	programs
Id_pod_prog	INT	ключевое поле	pod_ programs
Id_osn_meropr	INT	ключевое поле	osnovnoe_meropriyatie
Id_meropr	INT	ключевое поле	meropriyatie
data_plan	CHAR	плановая дата	
indikator	VARCHAR	название	
ed	INT	ключевое поле	ed
Indicator_znach	VARCHAR	значение	
visible	INT	логическое удаление	

Таблица 10 – описание таблицы муниципальных подпрограмм indicator_dost

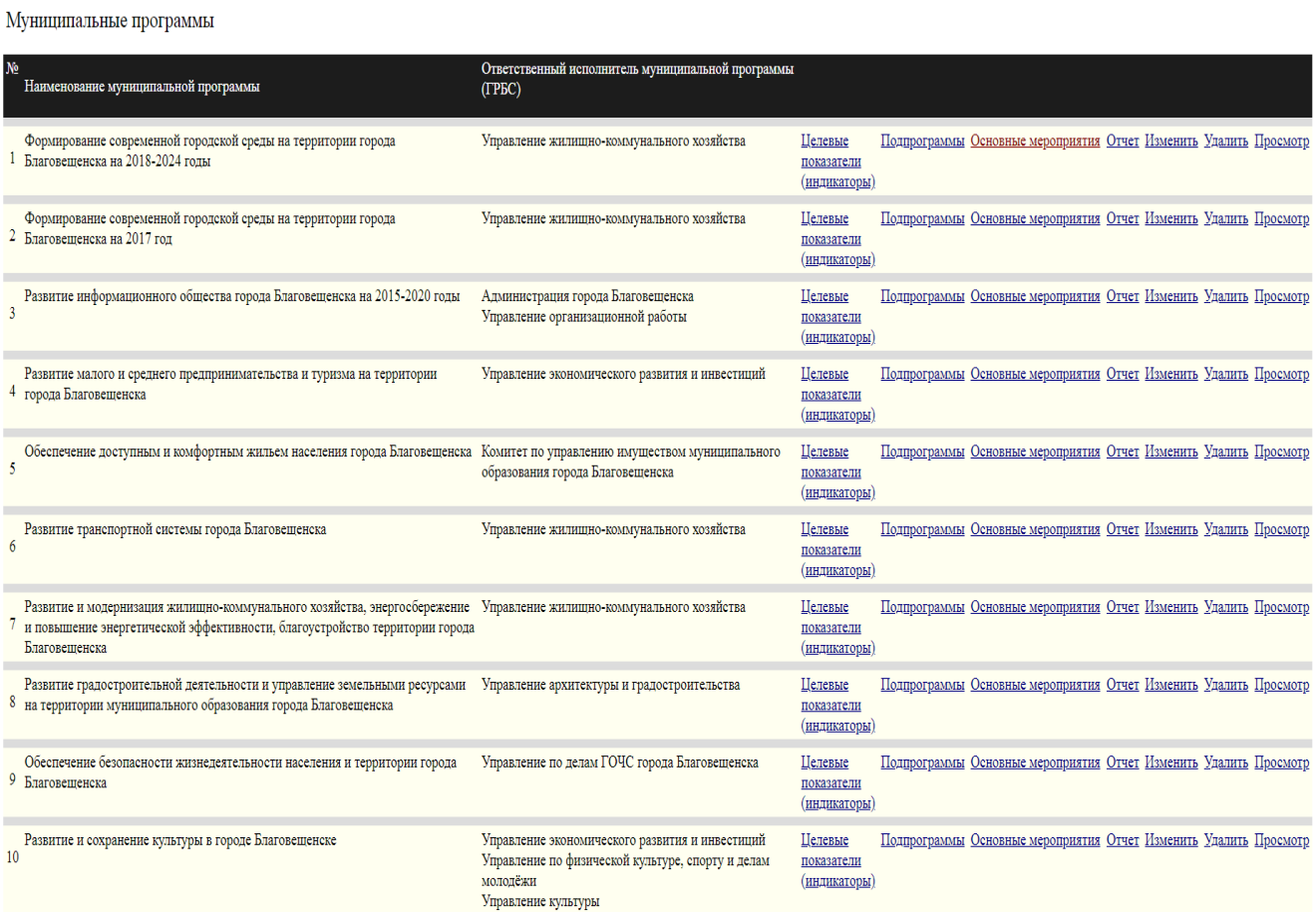
Имя поля	Тип данных	Комментарий	Ссылки на поле
id	INT	ключевое поле	
Id_prog	INT	ключевое поле	programs
Id_pod_prog	INT	ключевое поле	pod_ programs
Id_osn_meropr	INT	ключевое поле	osnovnoe_meropriyatie
Id_meropr	INT	ключевое поле	meropriyatie
id_indikator	INT	ключевое поле	indikator
period_id	INT	ключевое поле	period
ed	INT	ключевое поле	ed
period_name	VARCHAR	название периода динамическое	
data_plan	VARCHAR	дата	

Продолжение таблицы 10			
indikator_primechanie	VARCHAR	примечание	
indikator	VARCHAR	название динамическое	
Indicator_znach	VARCHAR	значение	
visible	INT	логическое удаление	

3.2 Описание интерфейса пользователя

Интерфейс пользователя разработанного программного обеспечения представляет собой множество форм близких к табличным, это связано с тем, что ответственные исполнители привыкли работать с таблицами и им проще воспринимать информацию в таком виде. Перед созданием электронных форм интерфейса были изготовлены эскизы форм и проведен опрос будущих пользователей. По результатам опроса были изготовлены электронные формы.

Самая первая форма этого программного обеспечения – форма авторизации, после авторизации пользователь попадает на главный экран (рисунок 16) имея свой уникальный набор прав. Если пользователю разрешено какое либо действие то кнопка подразумевающая это действие активна, иначе -она серая. Даже если пользователь нажмёт на такую кнопку, то никаких действий не произойдёт. Система присваивает глобальной переменной результат аутентификации пользователя и хранит его до закрытия программы или смены пользователя. Разграничение прав пользователей происходит в интерфейсе администратора. Интерфейс администратора описан ниже.


Муниципальные программы

№	Наименование муниципальной программы	Ответственный исполнитель муниципальной программы (ГРБС)						
1	Формирование современной городской среды на территории города Благовещенска на 2018-2024 годы	Управление жилищно-коммунального хозяйства	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
2	Формирование современной городской среды на территории города Благовещенска на 2017 год	Управление жилищно-коммунального хозяйства	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
3	Развитие информационного общества города Благовещенска на 2015-2020 годы	Администрация города Благовещенска Управление организационной работы	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
4	Развитие малого и среднего предпринимательства и туризма на территории города Благовещенска	Управление экономического развития и инвестиций	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
5	Обеспечение доступным и комфортным жильем населения города Благовещенска	Комитет по управлению имуществом муниципального образования города Благовещенска	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
6	Развитие транспортной системы города Благовещенска	Управление жилищно-коммунального хозяйства	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
7	Развитие и модернизация жилищно-коммунального хозяйства, энергосбережение и повышение энергетической эффективности, благоустройство территории города Благовещенска	Управление жилищно-коммунального хозяйства	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
8	Развитие градостроительной деятельности и управление земельными ресурсами на территории муниципального образования города Благовещенска	Управление архитектуры и градостроительства	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
9	Обеспечение безопасности жизнедеятельности населения и территории города Благовещенска	Управление по делам ГОЧС города Благовещенска	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр
10	Развитие и сохранение культуры в городе Благовещенске	Управление экономического развития и инвестиций Управление по физической культуре, спорту и делам молодёжи Управление культуры	Целевые показатели (индикаторы)	Подпрограммы	Основные мероприятия	Отчет	Изменить	Удалить	Просмотр

Рисунок 16 – Главный экран

Для создания новой муниципальной программы необходимо нажать соответствующую кнопку, затем ввести наименование, период действия, ответственных исполнителей и нажать на кнопку сохранить. После чего открываются все возможные варианты работы с муниципальной программой, появляется возможность создавать подпрограммы, основные мероприятия, мероприятия и заполнять финансирование по мероприятиям. В интерфейсе пользователя всегда доступны отчёты, их может создать любой пользователь прошедший идентификацию.

Более подробно интерфейс пользователя описан в инструкции пользователя, которая представлена на стартовой странице программного обеспечения и доступна без ввода имени пользователя и пароля.

3.3 Описание интерфейса администратора

Интерфейс администратора представлен формой с рядом функций:

- заполнение справочников;

- регистрация пользователей и назначение им прав;
- функции запрета редактирования всем;
- функция запрета удаления разделов всем;
- функция загрузки данных об исполнении бюджета в формате xml.

Назначение кнопок интуитивно понятно и не требует объяснений, более того, администратор системы как минимум должен обладать знаниями персонального компьютера и информационной системы на порядок выше обычного исполнителя. Пример назначения прав пользователю на редактирование и удаление представлен на рисунке 17.

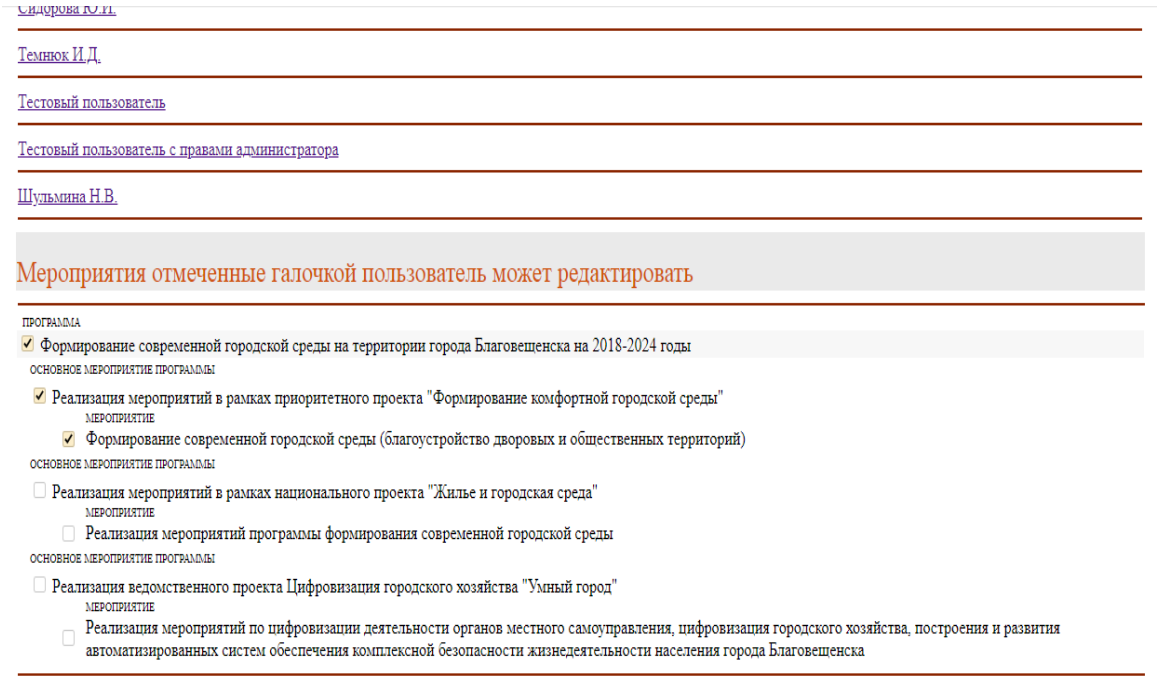


Рисунок 17 – Назначение прав пользователю

Переключение в интерфейс администратора происходит на уровне авторизации, то есть если у пользователя есть права администратора, то он имеет возможности обычного пользователя с просмотром и правкой муниципальный программ плюсом к этому кнопку перехода в интерфейс администратора.

3.4 Описание модуля формирования отчётов

Модуль формирования отчётов на сегодняшний день представлен четырьмя отчётами (рисунок 18), в дальнейшее планируется увеличение числа отчётов и разработка отчётов в виде графических диаграмм.

Отчеты

Отчет о ходе реализации муниципальной программы

Отчет о достижении целевых показателей (индикаторов) муниципальной программы

Отчет о достижении непосредственных результатах реализации основных мероприятий и мероприятий муниципальной программы

Результаты достижения показателей эффективности муниципальной программы

Рисунок 18 – Модуль «Отчёты»

Каждый отчёт имеет свои собственные парой даже уникальные настройки, так на пример наш огромный по содержанию отчёт с названием «Результаты достижения показателей эффективности муниципальной программы» в нашем варианте исполнения имеет обобщающее визуальное представление полезной информации о реализации муниципальных программ (рисунок 19) и более детальное (расширенное). При нажатии на кнопку «подробно» раскрывается список реализуемых или реализованных подпрограмм с процентом их реального выполнения.

Результаты достижения показателей эффективности муниципальной программы

Отчетный период 2019 год Создать отчёт

Наименование муниципальной программы (подпрограммы)		**Степень достижения, %**
Развитие физической культуры и спорта в городе Благовещенске	Подробнее	118,1
Развитие транспортной системы города Благовещенска	Подробнее	97,9
Развитие малого и среднего предпринимательства и туризма на территории города Благовещенска	Подробнее	58,2
Обеспечение безопасности жизнедеятельности населения и территории города Благовещенска	Подробнее	98,7
Обеспечение доступным и комфортным жильем населения города Благовещенска	Подробнее	87,5
Развитие градостроительной деятельности и управление земельными ресурсами на территории муниципального образования города Благовещенска	Подробнее	75,0
Развитие и модернизация жилищно-коммунального хозяйства, энергосбережение и повышение энергетической эффективности, благоустройство территории города Благовещенска	Подробнее	100,0
Развитие образования города Благовещенска	Подробнее	100,3
Развитие потенциала молодежи города Благовещенска	Подробнее	105,6
Развитие и сохранение культуры в городе Благовещенске	Подробнее	168,7
Формирование современной городской среды на территории города Благовещенска на 2018-2024 годы	Подробнее	100,0

Рисунок 19 – Общее представление информации

Как мы видим, процент выполнения может быть выше 100 процентов, это связано с тем, что в результате торгов цена контракта была сильно снижена и на образовавшуюся экономию были выполнены дополнительные однотипные работы. Например, вместо ремонта одного здания было отремонтировано два здания. Это грубый пример, но он отражает реальную ситуацию.

3.5 Ввод информационной системы в опытную эксплуатацию

Разработанное программное обеспечение в обязательном порядке должно доказать свою работоспособность.

Испытания вновь созданного программного обеспечения, как правило, представляют собой порой очень сложный процесс тщательной проверки правильного выполнения запрограммированных функций испытуемой системы по заранее подготовленному плану проверки, проверки на соответствие установленных показателей информационной системы требованиям технического задания, выявления проблем и устранения всех выявленных недостатков в реакциях системы на действия испытателя, в соответствии с разработанной документацией.

В соответствии с ГОСТ 34.603-92 "Информационная технология. Виды испытаний автоматизированных систем" для вновь созданных информационных систем, как правило, устанавливаются следующие испытания: предварительные, опытная эксплуатация, приемочные.

Предварительные испытания обязательно выполняют после тщательной проведения разработчиком процесса отладки программного комплекса и тестирования его технических средств испытуемой информационной системы и представления заказчику соответствующих регламентированных документов о полной готовности к проведению испытаний, а также после углубленного ознакомления всего заинтересованного персонала с эксплуатационной документацией. Проще говоря, с инструкциями пользователя.

Опытную эксплуатацию, как правило, проводят в целях определения фактических значений вновь созданной или модифицированной информационной системы и готовности ответственного персонала к работе с системой.

Разработанное программное обеспечение в данный момент прошло стадию испытания и принято в эксплуатацию. База данных наполнена реальными данными, сформированы и представлены руководству реальные отчёты. Отчёты в свою очередь выверены и представляют реальную картину по исполнению программных мероприятий администрации города Благовещенска (рисунок 20).

Синим цветом выделены ключевые поля, по которым осуществлена связь между таблицами базы данных. Зелёным цветом, собственно данные которые введены пользователями системы.

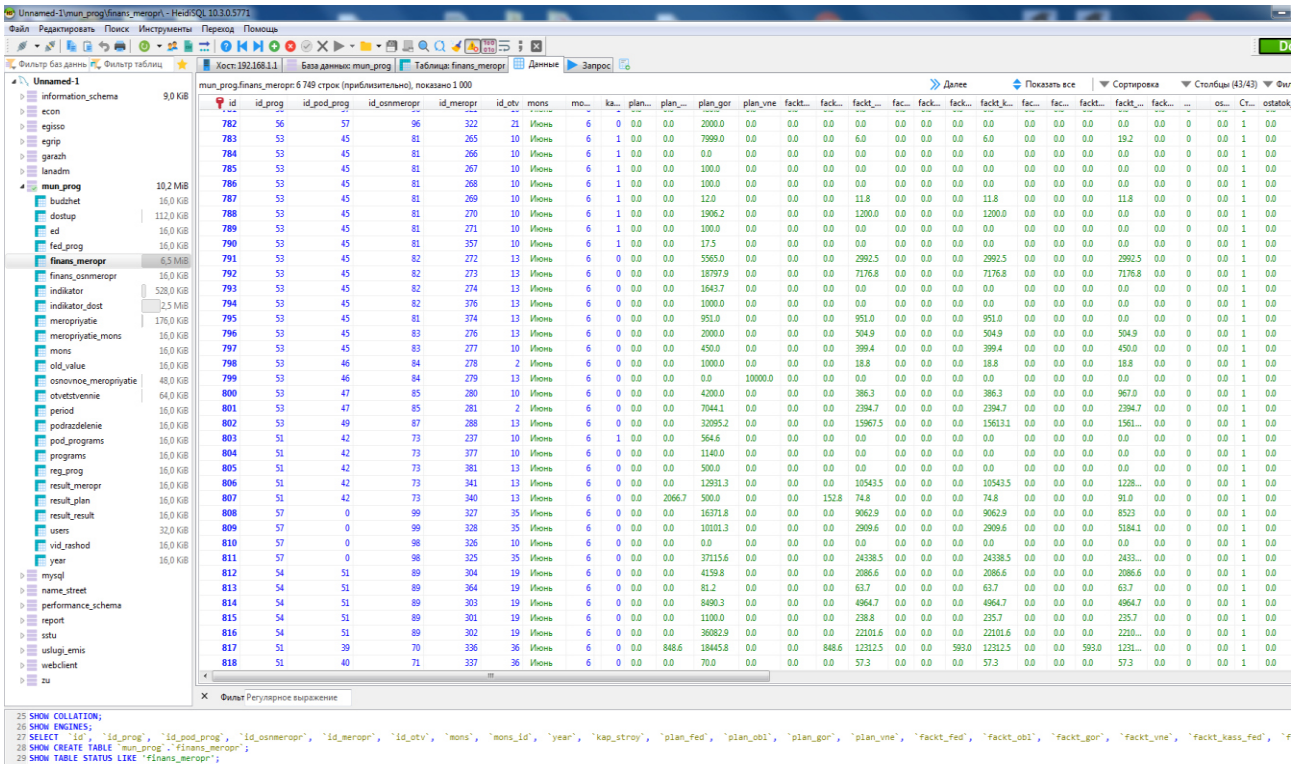


Рисунок 20 – Таблица базы данных

3.6 Выводы по главе

В данной главе представлено описание разработанного приложения. Приложение представляет из себя несколько компонентов: базу данных, интерфейс пользователя, интерфейс администратора и модуль отчётов. Описание компонентов системы является одним из способов наглядного отображения структуры программного обеспечения. Такое описание показывает ключевые моменты, без которых понимание логики ПО будет очень затруднено. Описание базы данных выполнено в виде логической структуры и демонстрирует логические связи между таблицами базы данных. Табличное представление базы данных демонстрирует связи между таблицами базы по ключевым полям. Взаимодействие (ввод и вывод данных) осуществляется с помощью запросов на языке SQL описанных в скриптах. Скрипты модулей и собственно классов, написанных на языке PHP. Взаимодействие модулей и классов разработанного программного обеспечения описано в виде диаграммы классов. Модуль формиро-

вания отчётов имеет табличное представление с автоматической подстановкой ответственного исполнителя по финансированию мероприятия. Модуль отчётов имеет потенциал к расширению, то есть в него возможно добавление новых форм отчётности при возникновении необходимости. В перспективе сюда будут добавлены отчёты в виде диаграмм различных типов (столбчатая, круговая и т.д.). Все описанные модули реализованы и введены в эксплуатацию. Описание экранных форм пользователя приведено не полностью в виду того, что подробное описание в данной работе не является принципиальным. Кроме того, более подробное описание пользовательских форм приведено в инструкции пользователя, которое является приложением к данной работе.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы создано программное обеспечение в виде веб-приложения в сети Интернет.

В ходе анализа предметной области, проектирования информационной системы, разработки программного приложения получены следующие результаты:

- исследована предметная область администрации;
- обоснована цель создания приложения;
- проанализированы, дополнены и оформлены требования к будущему приложению;
- рассмотрен комплекс технологий разработки и выбраны наиболее подходящие из них;
- выявлены функциональные и обеспечивающие подсистемы;
- разработан проект базы данных;
- спроектированы серверная и клиентская составляющие приложения;
- выполнена реализация базы данных в системе управления базы данных;
- согласно проектному описанию разработано приложение на языках программирования;
- описан пример работы системы;
- рассмотрены вопросы проектирования эргономичного графического интерфейса пользователя.

Таким образом, были выполнены все поставленные для данной выпускной квалификационной работы задачи.

Программное обеспечение разработано в соответствии с техническим заданием и успешно используется в администрации города Благовещенска.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Адамс, К. Администрирование сервера IIS 7 / К. Адамс. - Москва: Бином, 2010. - 362 с. - ISBN 978-5-9518-0367-2.

2. Архитектура клиент-сервер или Web: выбор разработчика [Электронный ресурс]. – Режим доступа: http://www.interface.ru/home.asp?artId=22674 – 25.05.2016.

3. Базовые CSS-стили различных элементов UML [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/155875/ - 24.10.2012.

4. Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/web-api/ - 07.04.2017.

5. Диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – Режим доступа: http://www.nnre.ru/kompyutery_i_internet/samouchitel_uml/p4.php - 18.03.2016.

6. Диаграмма деятельности и особенности ее построения [Электронный ресурс]. – Режим доступа: https://www.intuit.ru/studies/courses/32/32/lecture/1020 - 14.06.2019.

7. Диаграммы классов UML [Электронный ресурс]. – Режим доступа: https://pro-prof.com/archives/3212 - 07.06.2017.

8. Дюбуа, Поль MySQL: Пер. с англ. – Москва: Издательский дом «Вильямс», 2001. – 816 с.

9. Жизненный цикл тестирования ПО [Электронный ресурс]. – Режим доступа:http://ru.qatestlab.com/knowledge-center/qa-testing-materials/5-distinctions-between-a-client-server-and-web-application/ – 12.05.2017.

10. Знакомство с нотацией IDEF0 и пример использования [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/trinion/blog/322832/ - 28.02.2017.

11. Изучаем ASP.NET MVC 4 [Электронный ресурс]. - Режим доступа: http: //metanit.com/sharp/mvc/index.php.

12. Инструменты для рисования UML-диаграмм [Электронный ресурс]. – Режим доступа: http://msugvnua000.web710.discountasp.net/Posts/Details/3787 - 13.06.2010.

13. История развития интерфейсов [Электронный ресурс]. – Режим доступа: http://www.studfiles.ru/preview/1938111/page:2/ - 07.03.2015.

14. Каскадные таблицы стилей CSS [Электронный ресурс]. – Режим доступа: https://htmlweb.ru/css/styles.php - 26.05.2019.

15. Коггзолл, Джон PHP 5 Полное руководство: Пер. с англ. – Москва: Издательский дом «Вильямс», 2006 – 752 с.

16. Консалтинг в области информационных технологий (ИТ-консалтинг) построения [Электронный ресурс]. – Режим доступа: https://www.intuit.ru/studies/courses/532/388/lecture/9011?page=7 –14.06.2019.

17. Коржов, К. Многоуровневые системы клиент-сервер / К. Коржов. - Москва: Издательство «Открытые системы», 1997. - 208 с.

18. Кузнецов, М.В. Самоучитель PHP 5/6 / М. В. Кузнецов, И. В. Симдянов – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 672 с.: ил.

19. Мак-Дональд М. HTML5. Недостающее руководство: Пер. с англ. – СПб.: БХВ-Петербург, 2012. – 480 с.

20. Макфарланд Д. Большая книга CSS3. 3-е изд. – СПб.: Питер, 2014. – 608 с.

21. Мониторинг муниципальных программ [Электронный ресурс]. – Режим доступа: http://studik.n et/monitoring-municipalnyx-programm/ - 08.03.2012.

22. Основы UML. Диаграммы последовательности [Электронный ресурс]. – Режим доступа: https://pro-prof.com/archives/2769 - 19.07.2016.

23. Особенности проектирования интерфейсов информационных систем [Электронный ресурс]. – Режим доступа: https://revolution.allbest.ru/programming/00781099_0.html – 05.04.2017.

24. Паттерны проектирования на php [Электронный ресурс]. - Режим доступа: https://refactoring.guru/ru/design-patterns/php.

25. Проектирование баз данных программ [Электронный ресурс]. – Режим доступа: https://helpiks.org/5-40515.html - 22.08.205.

26. Проектирование информационной системы [Электронный ресурс]. – Режим доступа: https://finswin.com/projects/proektirovanie/informacionnyhsistem.html - 03.02.2017.

27. Проектирование программного обеспечения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/edison/blog/267569/ - 30.09.2015.

28. Разница между SQL и MySQL [Электронный ресурс]. - Режим доступа: https://refactoring.guru/ru/design-patterns/php.

29. Руководство по проектированию реляционных баз данных (1-3 часть из 15) [перевод] [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/193136/ - 10.09.2013.

30. Система управления базами данных mysql [Электронный ресурс]. – Режим доступа: https://depix.ru/articles/sistema_upravleniya_bazami_dannyh_mysql.

31. Технические требования к информационной системе [Электронный ресурс]. – Режим доступа: https://businessman.ru/tehnicheskie-trebovaniya-k-informatsionnoy-sisteme.html

32. Цели и задачи проектирования [Электронный ресурс]. – Режим доступа: http://msd.com.ua/chelovecheskij-faktor/celi-i-zadachi-proektirovaniya/ - 12.04.2018.

33. Что такое юзабилити или 7 основных правил юзабилити сайта [Электронный ресурс]. – Режим доступа: https://convertmonster.ru/blog/uzabiliti-blog/osnovy-juzabiliti-sajta/ - 25.02.2015.

34. Юзабилити веб-сайта: основы [Электронный ресурс]. – Режим доступа: https://www.maxstyle.ru/blog/yuzabiliti-veb-sayta-osnovy/ - 07.08.2016.

35. MS SQL и MySQL — что это и чем они отличаются [Электронный ресурс]. - Режим доступа: https://vchemraznica.ru/ms-sql-i-mysql-chto-eto-i-chem-oni-otlichayutsya/.

36. UML – быстрый старт [Электронный ресурс]. – Режим доступа: http://michaelsmirnov.blogspot.com/2011/03/uml.html – 01.02.2008.

ПРИЛОЖЕНИЕ А

Техническое задание.

1. Введение.

Работа выполняется в рамках проекта «Автоматизированная система мониторинга исполнения программных мероприятий администрации города Благовещенска».

1.2 Основание для разработки

Основанием для данной работы служит нормативные правовые акты РФ, администрации г. Благовещенска:

• бюджетный кодекс Российской Федерации;

• Федеральный закон от 28.06.2014 № 172-ФЗ «О стратегическом планировании в Российской Федерации»;

• Концепция повышения эффективности контрольно-надзорной деятельности органов государственной власти и органов местного самоуправления.

• распоряжение администрации города Благовещенска от 07.07.2015 № 117р «о разработке программного обеспечения для осуществления оперативного мониторинга исполнения программных мероприятий администрации города Благовещенска».

1.2. Наименование работы:

«Разработка программного обеспечения «Автоматизированная система мониторинга исполнения программных мероприятий администрации города Благовещенска».

1.3. Исполнитель: Шульга В.Н.

1.4. Соисполнители: нет.

3 Назначение разработки

Создание программного обеспечения для контроля и оперативного мониторинга состояния исполнения программных мероприятий администрации города Благовещенска.

4 Технические требования

4.1. Требования к функциональным характеристикам.

4.1.1. Состав выполняемых функций. Разрабатываемое программное обеспечение должно обеспечивать:

• сбор и анализ информации о поступлении и расходовании средств на реализацию программных мероприятий;

• вывод отчётных форм (форм много, можно указать в приложении) и визуализацию информации о текущей ситуации по исполнению ПМ;

• подготовку доклада мэру города;

• загрузку данных бюджета из файла экспорта поступившего из финансового управления.

4.1.2. Организация входных и выходных данных.

Исходные данные в систему поступают от пользователей системы, режим ввода, ручной.

Основной режим использования системы – еженедельная работа.

Вывод данных осуществляется в Интернет браузер в соответствии с заданными формами.

4.2. Требования к надежности.

Для обеспечения надежности функционирования системы необходим постоянный контроль введенных данных. Программная обработка вводимых пользователем данных, контроль итоговых сумм, введенных данных.

4.3. Условия эксплуатации и требования к составу и параметрам технических средств.

Для работы системы должен быть выделен ответственный сотрудник от каждого распорядителя бюджетных средств. Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

4.4. Требования к информационной и программной совместимости:

• программа должна работать на платформах семейства Windows 7 и выше;

• ПО должно иметь клиент-серверную архитектуру;

• ПО должно иметь WEB-интерфейс.

4.5. Требования к транспортировке и хранению. Программа поставляется на лазерном носителе информации.

Программная документация (руководство пользователя, руководство администратора) поставляется в электронном виде.

4.6. Специальные требования.

Программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации. В виду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов с ним.

Язык программирования – по выбору исполнителя, должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования.

5 Требования к программной документации

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой Системы Программной Документации (ЕСПД):

- руководство пользователя;
- руководство администратора;
- описание применения.

6 Технико-экономические показатели

Эффективность системы определяется удобством использования системы для контроля исполнения программных мероприятий администрации города Благовещенска и своевременного принятия решения управленческих решений.

7 Порядок контроля и приемки

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

ПРИЛОЖЕНИЕ Б

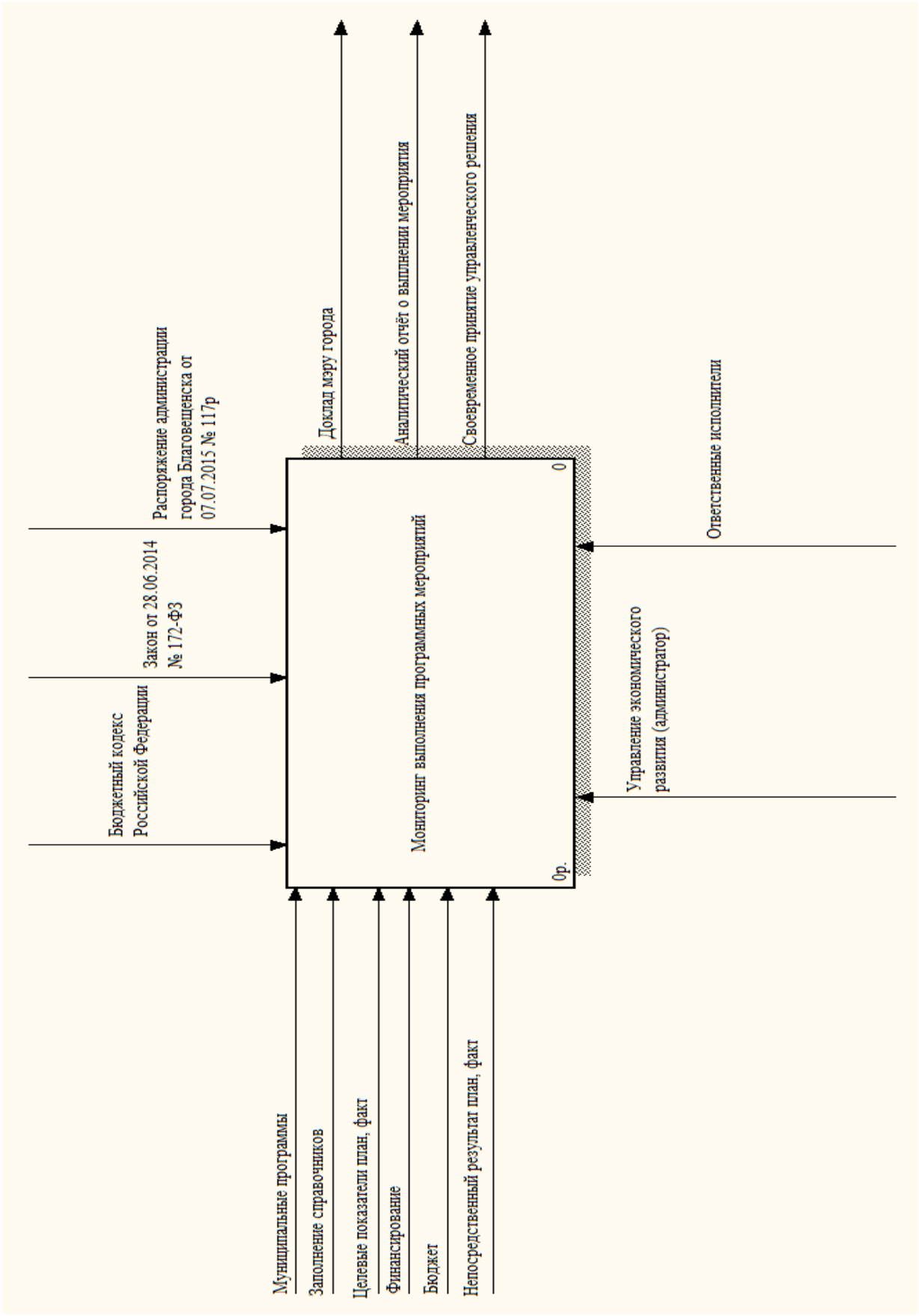


Рисунок Б.1 – Диаграмма предметная область в нотации IDFE0 A-0

ПРИЛОЖЕНИЕ В

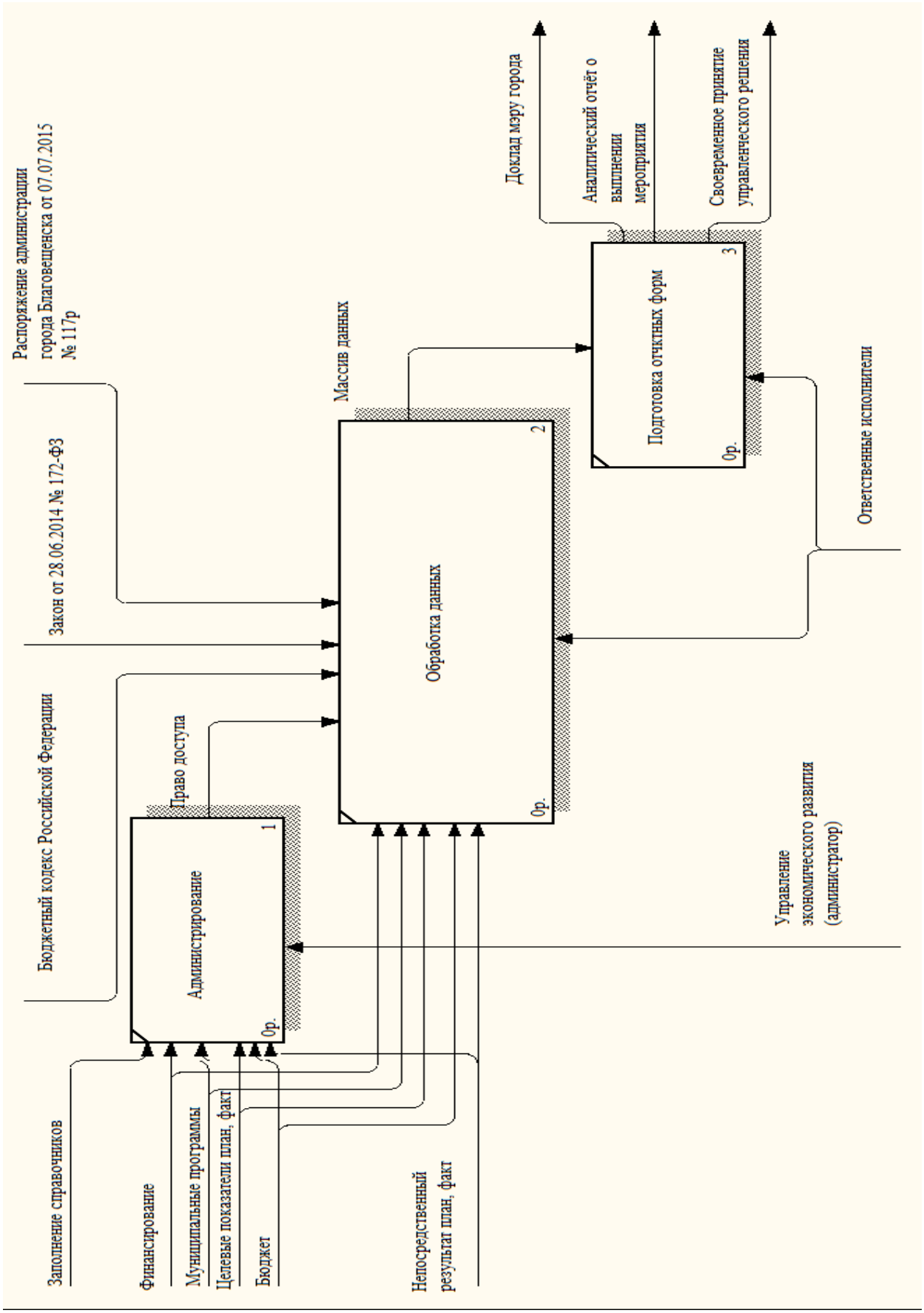


Рисунок В.1 – Диаграмма предметная область в нотации IDFE0 A0

ПРИЛОЖЕНИЕ Г

Рисунок Г.1 – Диаграмма базы данных стандарта UML

ПРИЛОЖЕНИЕ Д

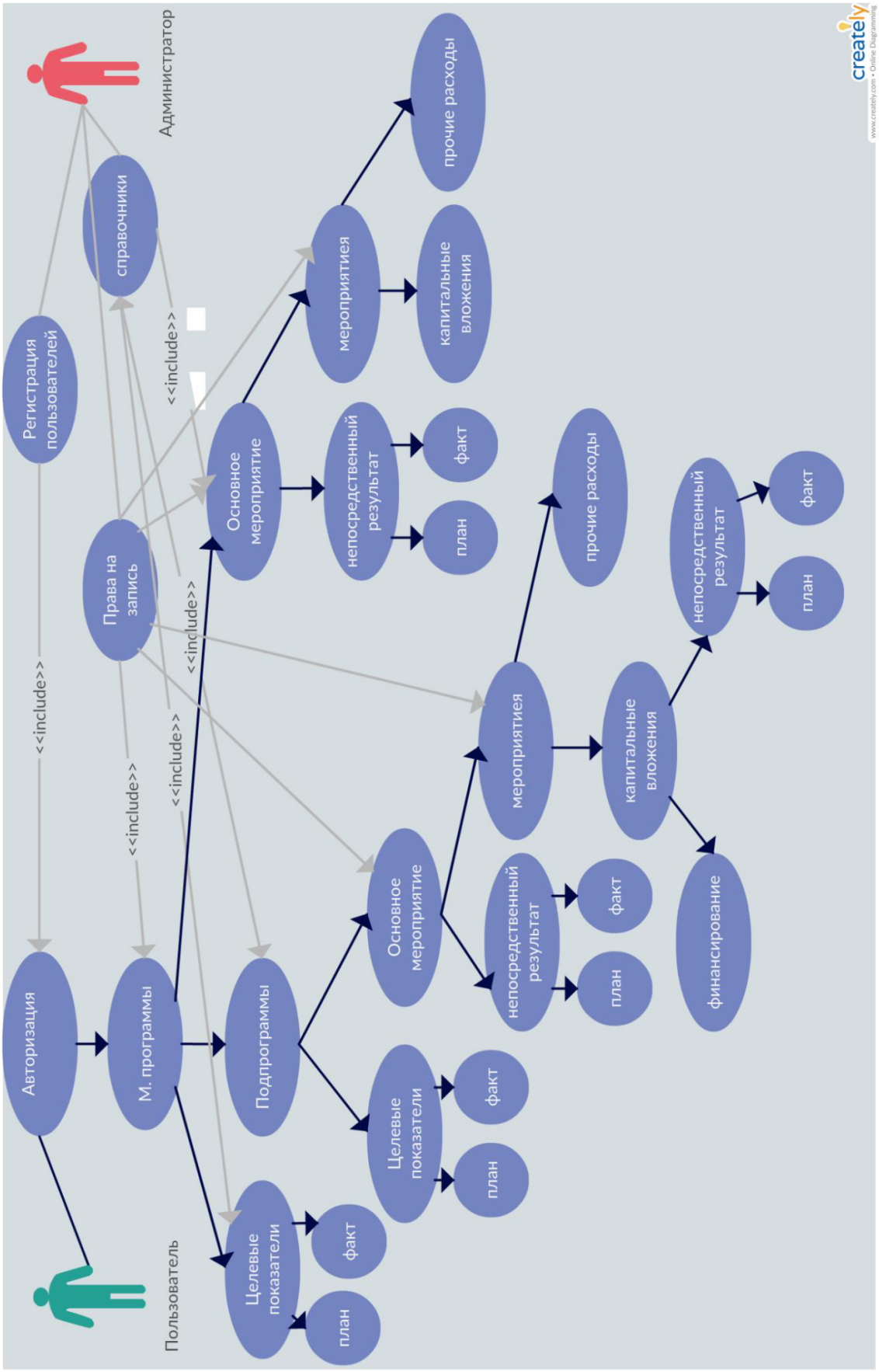


Рисунок Д.1 – Диаграмма прецедентов

ПРИЛОЖЕНИЕ Е

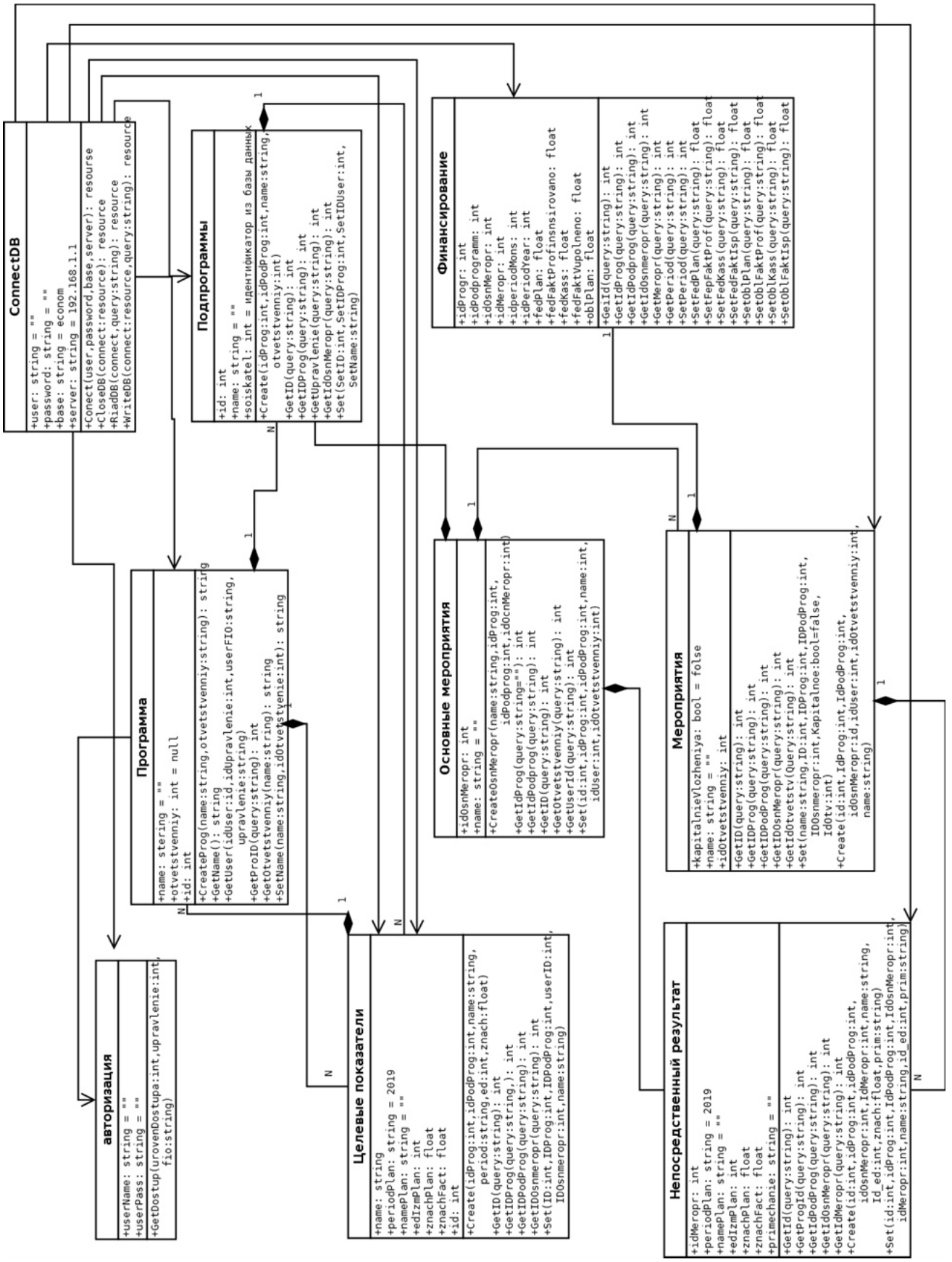


Рисунок Е.1 – Диаграмма классов стандарта UML