

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Дисциплина «Объектно-ориентированное программирование»

«К ЗАЩИТЕ ДОПУСТИТЬ»

Руководитель курсового проекта
старший преподаватель

_____._____.2021 Т.М. Унучек

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**«УПРАВЛЕНИЕ ИТ-АКТИВАМИ ПРЕДПРИЯТИЯ И ЕГО
ПРОГРАММНАЯ ПОДДЕРЖКА»**

БГУИР КП 1-40 05 01-10 012 ПЗ

Выполнила студент группы 914301
ЕРЁМЕНКО Николай Владимирович

(подпись студентки)

Курсовой проект представлен на
проверку 01.12.2021

(подпись студентки)

Минск 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ОПИСАНИЕ ПРОЦЕССА УПРАВЛЕНИЯ АКТИВАМИ.....	8
1.1 Общие понятия и термины	8
2 ПОСТАНОВКА ЗАДАЧИ УПРАВЛЕНИЯ ИТ-АКТИВАМИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ	11
3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0.....	14
4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ	20
5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ.....	22
6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ.....	25
6.1 Диаграмма классов.....	25
6.2 Диаграмма компонентов.....	27
6.3 Диаграмма развертывания.....	27
6.4 Диаграмма состояний	29
6.5 Диаграмма последовательности	30
6.6 Диаграмма вариантов использования	32
7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	34
8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ИТ-АКТИВАМИ	42
ЗАКЛЮЧЕНИЕ.....	46
Список использованных источников.....	47
ПРИЛОЖЕНИЕ А.....	48
ПРИЛОЖЕНИЕ Б	52
ПРИЛОЖЕНИЕ В.....	54
ПРИЛОЖЕНИЕ Г.....	63

ВВЕДЕНИЕ

Информационные технологии в современном мире интенсивно развиваются и применяются повсеместно. Наиболее широко применяемые средства в наше время – это сотовая связь и интернет, мобильные телефоны и компьютеры. Тем не менее, каждая отрасль науки и производства имеет своё специфическое оборудование, а также специально разработанное программное обеспечение, позволяющее выполнять огромный перечень задач наиболее эффективным и быстрым способом.

Рациональное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными и человеческими ресурсами. Следовательно, повышение эффективности становится одним из направлений модернизации деятельности в целом. Одним из самых очевидных и применяемых способов повышения эффективности осуществления различных процессов является их автоматизация. Стремительное развитие информационных компьютерных технологий, совершенствование технической платформы и появление принципиально новых классов программных продуктов привело к изменению подходов к автоматизации управления предприятиями.

Автоматизация – это не попытка заменить людей бездушными машинами, на практике это наоборот способ дать людям возможность работать эффективнее, быстрее и фокусироваться на интересных творческих задачах, принося больше пользы.

Перенос рутинных, зачастую выполняемых вручную, операций в специализированные системы позволяет компаниям экономить свои ресурсы, снижать количество ошибок и выполнять задачи быстрее и более качественно.

Управление активами – одна из важнейших стратегических областей деятельности организации. Одним из ключевых факторов управления активами, является возможность проведения инвентаризации, выявляющей детальную информацию по необходимому аппаратному и программному обеспечению, которая позволяет анализировать имеющиеся данные по определенным параметрам.

Сам по себе процесс управления активами в большинстве случаев не подразумевает никаких специфических трудностей, однако на практике является довольно однообразным, поскольку включает в себя множество повторяющихся элементов и действий, которые необходимо выполнять с определенным временным промежутком.

Организации, которые разработали, внедрили и поддерживают программу управления активами минимизируют риски и прочие затраты, которые могут возникнуть, если не использовать современные технологии для оптимизации системы.

Для малого и среднего бизнеса, практика управления активами может быть не настолько охватывающей и влияющей на многие отделы организации. Однако активы также необходимо отслеживать, управлять ими, выполнять необходимые финансовые расчеты по использованию ресурсов и распределению бюджета. Практика управления активами для маленьких компаний должна позволять получать необходимую информацию об активах, которыми организация владеет, управлять онлайн и оффлайн активами, а также планировать будущее развитие с учётом предполагаемых финансовых затрат. Все задачи для управления активами должны решаться эффективно, своевременно, с учётом задействованных трудовых ресурсов персонала.

Одним из главных результатов автоматизации является освобождение персонала от однообразной работы. Программа управления активами позволяет формировать различные отчеты и вести аналитический учет.

Таким образом, целью данного курсового проекта является сокращение временных затрат на управление активами за счёт автоматизации обработки необходимой информации: различных расчётов, процесса составления отчетности, визуализация данных в виде диаграмм.

Для достижения поставленной цели требуется решить следующие задачи:

- проанализировать заданную предметную область;
- рассмотреть физическую и логическую модель представления данных;
- создать базу данных основных активов и финансовых показателей;
- реализовать серверную часть, имеющую доступ к базе данных и выполняющую ряд необходимых действий, позволяющих достигнуть цель курсового проекта;
- реализовать клиентскую часть с интуитивно понятным интерфейсом, обеспечивающим максимально удобное взаимодействие с программой.
- протестировать разработанное приложение;
- разработать и описать руководство пользователя;
- разработать диаграмму состояний;
- разработать диаграммы последовательности;
- разработать диаграмму классов;
- разработать диаграмму компонентов;
- разработать диаграмму вариантов использования.

1 ОПИСАНИЕ ПРОЦЕССА УПРАВЛЕНИЯ АКТИВАМИ

1.1 Общие понятия и термины

На первом этапе проектирования информационной системы необходимо выполнить анализ предметной области, т.е. определить объекты предметной области и связи между объектами.

При выборе состава и структуры предметной области возможны два подхода: функциональный и предметный.

Функциональный подход реализует принцип движения «от задач» и применяется, когда определен комплекс задач, для обслуживания которых создается информационная система. В этом случае можно выделить минимальный необходимый набор объектов предметной области, которые должны быть описаны.

В предметном подходе объекты предметной области определяются с таким расчетом, чтобы их можно было использовать при решении множества разнообразных, заранее не определенных задач. Чаще всего используется комбинация этих двух подходов.

Активы — это программное обеспечение, оборудование и любые другие физические единицы, которыми владеет компания. Чаще всего активы относят к капитальным затратам [2].

Оборудование, являющееся активами, может включать:

- компьютеры;
- сети;
- серверы;
- периферийные устройства;
- телефоны;
- мобильные устройства;
- приложения, установленные на компьютер;

В зависимости от рода деятельности компании, могут быть и другие виды физических активов, такие как принтеры, инструменты, то есть все, чем владеет компания и предоставляет своим сотрудникам или клиентам.

Программные активы могут включать:

- лицензии на программное обеспечение;
- различные версии программного обеспечения;
- приложения, установленные на компьютер.

Важно не только знать о своих активах, но и управлять ими, то есть осуществлять различные финансовые и иные расчеты для грамотного распределения ресурсов. Еще несколько лет назад компании, которые занимались управлением активов, делали это вручную на бумаге или в таблицах на компьютере и называли такую базу портфелем активов. Несмотря на то, что электронные таблицы поддерживают формулы и макросы, их использование связано с ошибками, не дает высокой точности, требует много времени для ведения и в целом морально устарело.

Развитие технологий и применение облачных вычислений позволяют современным системам управления ИТ-активами быстрее и точнее отслеживать их показатели и производить необходимые расчеты.

Управление ИТ активами позволяет специалистам и управленцам в сфере информационных технологий вести разговор с бизнесом «на языке денег». Это позволяет не только понять друг друга, но и совместить стратегии развития бизнеса и информационных технологий в компании. А это нетривиальная и критически важная задача для любой компании [3].

Для первоначальной оценки актив, изначально необходимо определить стоимость по которой они были приобретены или иные затраты, которые компания вложила в их реализацию. Существует огромное количество различных видов активов, имеющих определенные сроки использования. Данную закономерность можно легко представить на примере программных средств, взяв за основу лицензионные версии программного обеспечения.

Для выполнения необходимых задач и реализации основной деятельности компании по производству информационных технологий, необходимо иметь соответствующее оборудование и программное обеспечение, с помощью которого будет создаваться определенный продукт. Для легального коммерческого использования некоторых средств разработки, необходимо иметь лицензию, которая также в большинстве случаев расширяет доступный для использования функционал.

Поскольку лицензионные соглашения для программного обеспечения имеют сроки использования, необходимо своевременно актуализировать информацию. Аналогичные действия нужны и для остальных активов. Ряд оборудования, которым обладает предприятие, имеет определенную ценность, а также со временем изнашивается и требует вложения средств на оптимизацию.

Первоначальная стоимость – фактическая стоимость приобретения активов, которая включает все затраты. Компаниям важно знать конкретные цифры затрат, для расчета доли бюджета, которая выделяется на приобретение всех активов.

Срок полезного использования – это период, в течение которого использование активов предприятия должно служить для выполнения целей ее деятельности. Например, срок действия лицензии на программное обеспечение.

Физический износ происходит в результате активной работы оборудования, а также под влиянием иных факторов.

Имея перечень представленных данных, можно определять и другие показатели, например, высчитывать износ актива за определенный промежуток времени, чтобы иметь представления, когда необходимо проводить обновление, помимо этого в компаниях рассчитываются финансовые показатели, которые позволяют определить годовые отчисления из бюджета на приобретение новых активов.

Используя некоторые активы, предприятие получает определенные выгоды и формирует прибыль, которая входит в основные финансовые показатели. На основе этих данных, можно определять следующие показатели активов:

- окупаемость актива;
- рентабельность актива;
- суммарная выгода от актива.

Прибыль нельзя считать единственным критерием успешности инициативы, поскольку равную норму доходности могут показать предприятия с совершенно разными показателями сумм оборотных средств и основных фондов. Следовательно, более эффективным может считаться то предприятие, у которого меньшая стоимость производственных фондов.

Таким образом, можно прийти к выводу, что прибыльность является абсолютным показателем. Рентабельность проекта – относительный показатель, характеризующий экономическую эффективность использования всех имеющихся активов [5].

Окупаемость — показатель эффективности, исчисляемый как соотношение затрат и прибыли.

Рентабельность – относительный показатель экономической эффективности. Коэффициент рентабельности рассчитывается как отношение прибыли к стоимости.

Основной и первостепенной целью разработки и внедрения процессов управления активами является обеспечение прозрачности и учета информационных активов, а также обеспечение финансовой эффективности их использования. Вторая цель достигается как при помощи грамотного управления лицензионными политиками и оптимизацией закупок, так и с помощью полноценного контроля за всеми ИТ активами в компании, их использованием и потребностями предприятий в новых и уже приобретенных активах.

Важной задачей является верный расчёт всех показателей для каждого актива. Даже самый грамотный специалист под воздействием человеческого фактора может допустить ошибку, которая может привести к лишним затратам ресурсов. Автоматизация управления активами уменьшит временные затраты и человеческие ресурсы [14].

Необходимо создать программный продукт, который будет отвечать следующим требованиям:

- должна быть создана база данных для быстрого и удобного поиска интересующей информации;
- сотрудник должен затрачивать минимальное количество временных ресурсов на различные расчёты и составление отчётов и графиков;
- интерфейс должен быть простым и интуитивно понятным для эффективной работы.

2 ПОСТАНОВКА ЗАДАЧИ УПРАВЛЕНИЯ ИТ-АКТИВАМИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ

В данном курсовом проекте поставлена цель сократить ресурсы и автоматизировать систему управления активами предприятия. Для этого необходимо разработать такое приложение, которое предоставит возможность многим пользователям вносить и просматривать необходимую информацию.

Предлагается разработать приложение в архитектуре клиент-сервер. Реализация серверного приложения будет осуществлена в виде консольного приложения, отображающая информацию о подключениях и запросах от пользователя. Для реализации данного проекта необходимо выполнить следующие задачи:

- изучить предметную область;
- реализовать клиент-серверное приложение;
- реализовать авторизацию пользователя;
- реализовать регистрацию пользователя;
- авторизацию администратора;
- регистрацию администратора;
- просмотр и корректировка активов предприятия;
- расчет основных показателей приведенных активов;
- добавление новых данных;
- удаление записей из базы данных;
- построение графиков на основе рассчитанных данных;
- формирование текстового отчёта.

Для максимального удобства клиентское приложение должно быть реализовано в виде оконного приложения с графическим интерфейсом пользователя. Функционал администратора отличается от функционала рядового пользователя, наличием ряда дополнительных возможностей.

Для реализации поставленной задачи используется объектно-ориентированный язык Java. При создании серверных приложений с помощью Java встречается минимальное число проблем и наиболее ярко проявляются достоинства Java-технологии. Здесь присутствует удобный механизм многопоточности, который облегчает создание приложений, обслуживающих одновременно множество пользователей. Важный фактор – широкая поддержка серверных Java-технологий всеми заметными производителями СУБД, серверов приложений и мониторов транзакций

Для реализации работы с базами данных была выбрана СУБД MySQL. База данных MySQL – это самая популярная в мире база данных с открытым кодом. MySQL является очень быстрым, надежным и легким в использовании. ПО MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

Графический интерфейс клиентской части реализуется с помощью платформы JavaFX. Она предоставляет большие возможности по сравнению с рядом других подобных платформ. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое. Для визуального редактирования файлов пользовательского интерфейса будет использоваться редактор Scene Builder. Интерфейс создаётся с помощью файлов разметки FXML. Этот способ хорошо подходит для отделения контроллеров от представлений, а использование Scene Builder позволит избежать прямой работы с данными.

Соединение между серверной и клиентскими частями должно устанавливаться с помощью протокола TCP/IP. Этот протокол обладает одним важным преимуществом: он обеспечивает аппаратную независимость. Так как в сетевых протоколах определяется только блок передачи и способ его отправки, TCP/IP не зависит от особенностей сетевого аппаратного обеспечения, позволяя организовать обмен информацией между сетями с различной технологией передачи данных. Система IP-адресов, в свою очередь, позволяет без затруднений установить соединение между любыми двумя машинами сети. Взаимодействие между устройствами в рамках стека TCP/IP осуществляется с помощью связки IP-адреса и номера порта.

Эта связка образует сокет – программный интерфейс, который обеспечивает обмен данными между устройствами на низком уровне. Протокол TCP/IP основывается на соединениях, устанавливаемых между двумя компьютерами, обычно называемых клиентом и сервером. Поэтому различают клиентский сокет и серверный сокет. Для организации взаимодействия клиент должен знать IP-адрес и номер порта сервера, по которым он осуществляет подключение к удаленному устройству.

Для хранения информации используется система управления реляционными базами данных MySQL. В реляционной базе данных информация хранится в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. Для упрощения работы с сервером MySQL используется MySQL Workbench.

Описанные технологии и программные средства характеризуются в первую очередь надёжностью и простотой использования. Совместное их применение позволит создать удобное многофункциональное приложение, отвечающее всем поставленным требованиям.

В разработке программного обеспечения паттерн проектирования программного обеспечения – это общее, многократно используемое решение часто встречающейся проблемы в данном контексте при разработке программного обеспечения. Это не законченный проект, который может быть преобразован непосредственно в исходный или машинный код. Скорее, это описание или шаблон для решения проблемы, которые можно использовать в самых разных ситуациях. Паттерны проектирования - это формализованные передовые практики, которые программист может использовать для решения общих проблем при разработке приложения или системы.

Паттерны объектно-ориентированного проектирования обычно показывают отношения и взаимодействия между классами или объектами без указания конечных классов приложений или объектов, которые в них участвуют. Паттерны, которые подразумевают изменяемое состояние, могут не подходить для функциональных языков программирования, некоторые паттерны могут быть лишними в языках, которые имеют встроенную поддержку для решения проблемы, которую они пытаются решить, и объектно-ориентированные паттерны не обязательно подходят для не объектно-ориентированных языков.

В паттерне "Модель – представление – контроллер" модель представляет данные приложения и связанную с ними бизнес-логику. Модель может быть представлена одним объектом или сложным графом связанных объектов.

Представление – это наглядное отображение содержащихся в модели данных. Подмножество модели содержится в отдельном представлении, таким образом, представление действует в качестве фильтра для данных модели. Пользователь взаимодействует с данными модели с помощью предлагаемого представлением наглядного отображения и обращается к бизнес логике, которая, в свою очередь, воздействует на данные модели.

Контроллер связывает представление с моделью и управляет потоками данных приложения. Он выбирает, какое представление визуализировать для пользователя в ответ на вводимые им данные и в соответствии с выполняемой бизнес-логикой. Контроллер получает сообщение от представления и пересылает его модели. Модель, в свою очередь, подготавливает ответ и отправляет обратно контроллеру, где происходит выбор представления и отправка пользователю.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

Методология функционального моделирования IDEF0 – это технология описания системы в целом как множества взаимозависимых действий. Важно отметить функциональную направленность IDEF0 – функции системы исследуются независимо от объектов, которые обеспечивают их выполнение. IDEF0 сочетает в себе небольшую по объему графическую со строгими и четко определенными рекомендациями, в совокупности предназначенными для построения качественной и понятной модели системы.

Каждая модель должна иметь контекстную диаграмму верхнего уровня, на которой объект моделирования представлен единственным блоком с граничными стрелками. Эта диаграмма называется А-0. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой.

Модель состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы – главные компоненты модели, все функции и интерфейсы на них представлены как блоки и дуги. Место соединения дуги с блоком определяет тип интерфейса:

Тип интерфейса:

- управляющая информация входит в блок сверху;
- входная информация входит в блок слева;
- результаты выходят из блока справа;
- механизм (человек или автоматизированная система), который осуществляет операцию, входит в блок снизу.

Каждый компонент модели может быть декомпозирован (расшифрован более подробно) на другой диаграмме. Рекомендуется прекращать моделирование, когда уровень детализации модели удовлетворяет ее цель. Общее число уровней в модели не должно превышать 5-6.

Построение диаграмм начинается с представления всей системы в виде одного блока и дуг, изображающих интерфейсы с функциями вне системы. Затем блок, который представляет систему в качестве единого модуля, детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Каждая детальная диаграмма является декомпозицией блока из диаграммы предыдущего уровня.

Наиболее важные свойства объекта выявляются на верхнем уровне иерархии, по мере декомпозиции функции верхнего уровня и разбиения ее на подфункции эти свойства уточняются. Каждая подфункция декомпозируется на элементы следующего уровня. Декомпозиция происходит до тех пор, пока не будет получена релевантная структура, позволяющая ответить на вопросы, сформулированные в цели моделирования.

Контекстная диаграмма представлена на рисунке 3.1

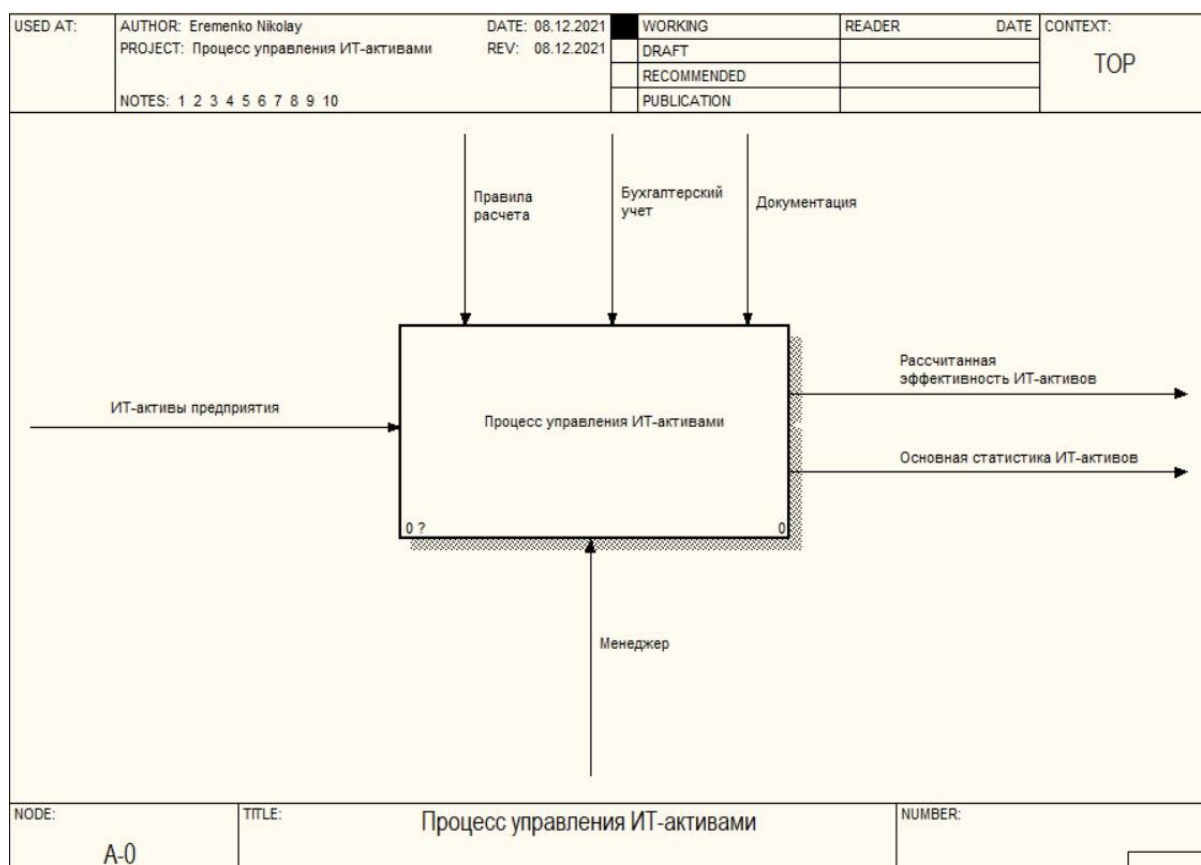


Рисунок 3.1 – Контекстная диаграмма верхнего уровня

Управление процессом осуществляется с использованием норм бухгалтерского учета, правил расчетов, а также документации. Бухгалтерский учет и правила расчетов необходимы, поскольку данные нужно фиксировать по определенным критериям, заполняя соответствующие документы. Сотрудник, который взаимодействует с информацией, указан снизу, это менеджер компании, который руководит данным процессом и следит за его выполнением, а также несет ответственность. На входе расположены ИТ-активы предприятия, их закупают по мере необходимости и в дальнейшем используют для реализации поставленных задач. Результатом выполнения процесса являются расчеты эффективности ИТ-активов и основная статистика.

На следующем рисунке 3.2 представлена декомпозиция процесса управления ИТ-активами.

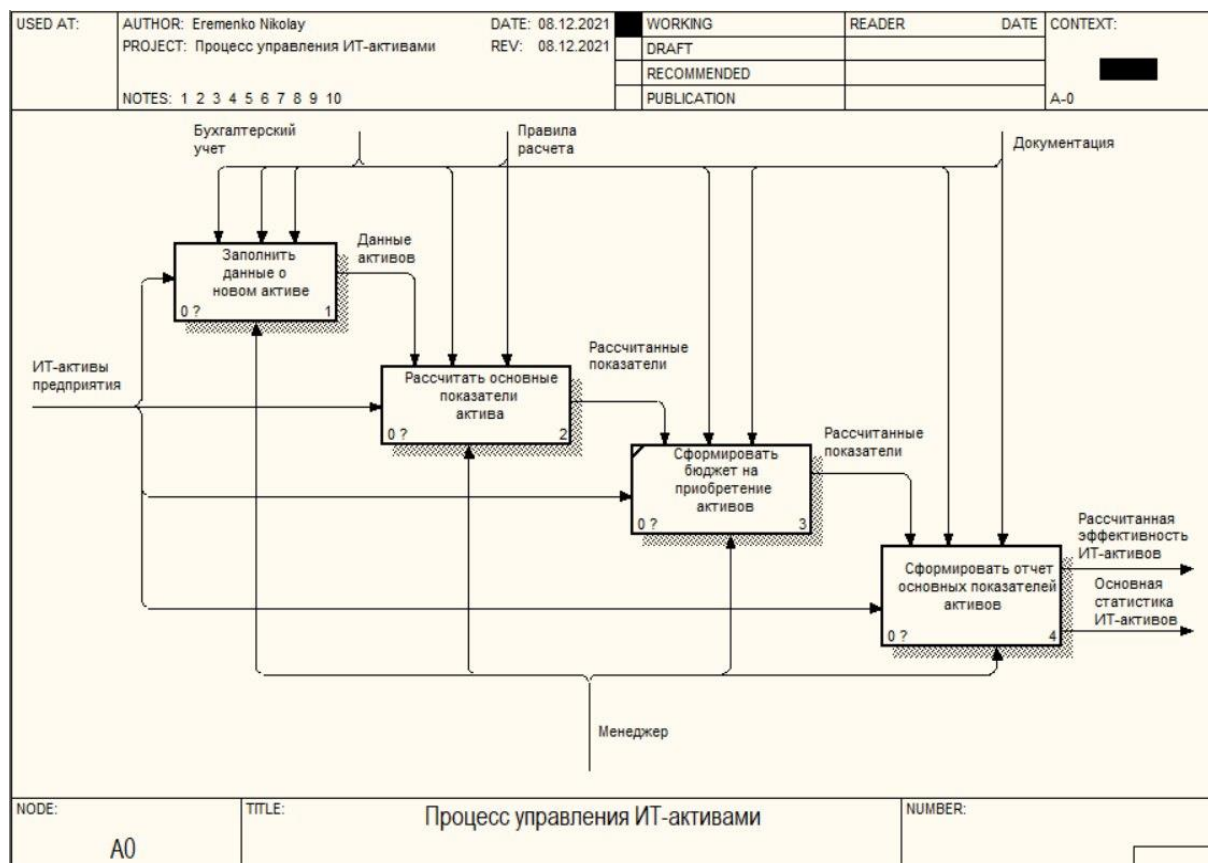


Рисунок 3.2 – Декомпозиция контекстной диаграммы верхнего уровня

На диаграмме представлены следующие блоки:

- Заполнение данных о новом активе;
- Расчет основных показателей актива;
- Формирование бюджета на приобретение активов;
- Формирование отчета основных показателей активов.

Процесс начинается с заполнения основных данных о новом активе. На этом этапе менеджер заносит в систему всю необходимую для дальнейших расчётов информацию. После того, как основная информация занесена в систему, происходит автоматизированный расчет основных показателей, на основе которых можно формировать бюджет предприятия, а также в дальнейшем формировать отчет всех основных показателей для анализа их эффективности. Этот процесс является ключевым, поскольку каждому предприятию необходимо производить долгосрочное планирование и распределять свой бюджет заранее.

Декомпозиция блока «Заполнения данных о новом активе» представлена на рисунке 3.3.

Декомпозиция блока «Заполнения данных о новом активе» содержит четыре блока:

- Ввод данных о наименовании актива;
- Ввод данных о первоначальной стоимости актива;
- Ввод данных о сроках годности актива;
- Анализ внесенной информации об активах.

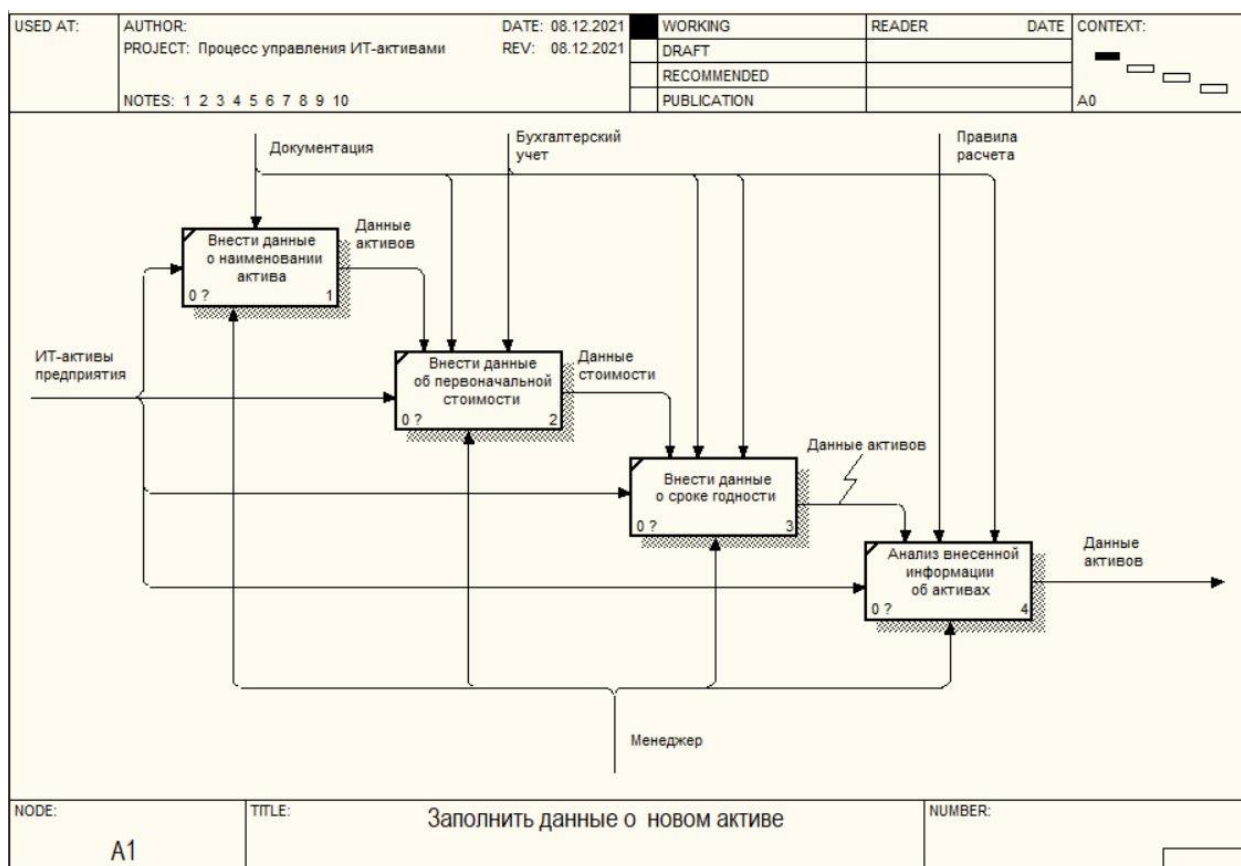


Рисунок 3.3 – Декомпозиция блока «Заполнение данных об активе»

На данном этапе менеджер заполняет основную информацию об активах, изначально он вносит наименование актива, после чего данные об первоначальной стоимости, как было описано раньше, все данные параллельно вносятся в документацию, которая соответствует определенным правилам. Поскольку у каждого актива есть свой срок годности, т.е. временной промежуток, в течение которого актив можно использовать, менеджер вносит эту информацию, далее данные сохраняются в системе и при необходимости их можно обрабатывать и представлять в нужном формате.

Декомпозиция блока расчета основных показателей отображена на рисунке 3.4

На ней представлены следующие блоки:

- получение данных;
- расчет окупаемости;
- расчет рентабельности;
- расчет суммарной выгоды.

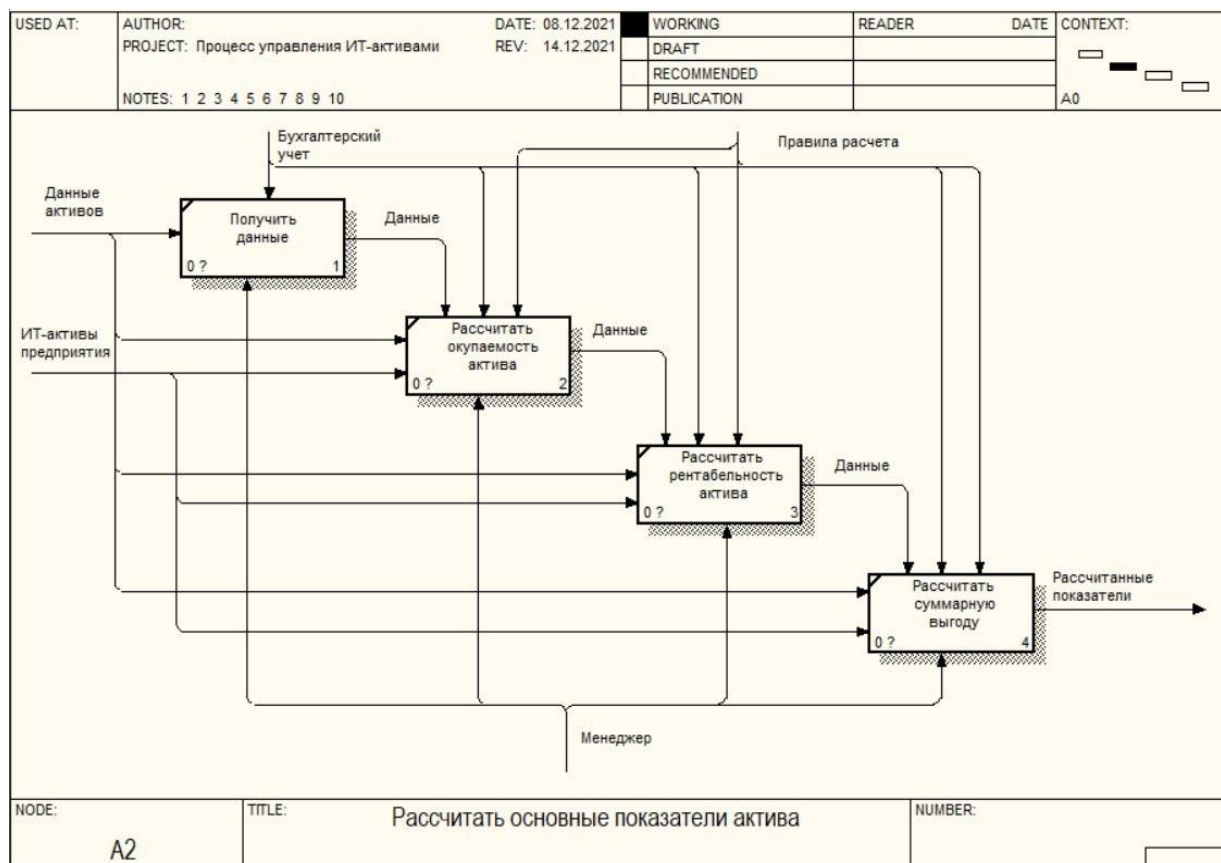


Рисунок 3.4 – Декомпозиция блока «Расчет основных показателей»

Изначально сюда поступают первоначальные данные об активах, которые менеджер внес в систему, после получения и анализа данных, изначально рассчитывается окупаемость актива, далее на основе всех данных, рассчитывается рентабельность актива и в заключении анализируется планируемая суммарная выгода от актива за время его использования. При этом при расчете общей выгоды от актива, не учитываются возможные затраты, которые могут возникнуть в ходе эксплуатации.

Рассмотрим более подробно «Формирование отчета основных показателей активов». На данном этапе, мы получаем рассчитанные ранее данные, которые можно обрабатывать и на их основе формировать бюджет предприятия. Изначально данные анализируются, после чего для более удобного просмотра всех данных, они визуализируются на диаграммах по определенным критериям. На последнем этапе формируется отчет об активах, который сохраняется в необходимом формате.

Декомпозиция этого блока представлена на рисунке 3.5.

Она содержит следующие блоки:

- Анализ рассчитанных данных;
- Визуализация данных на диаграммах;
- Формирование отчета об активах.

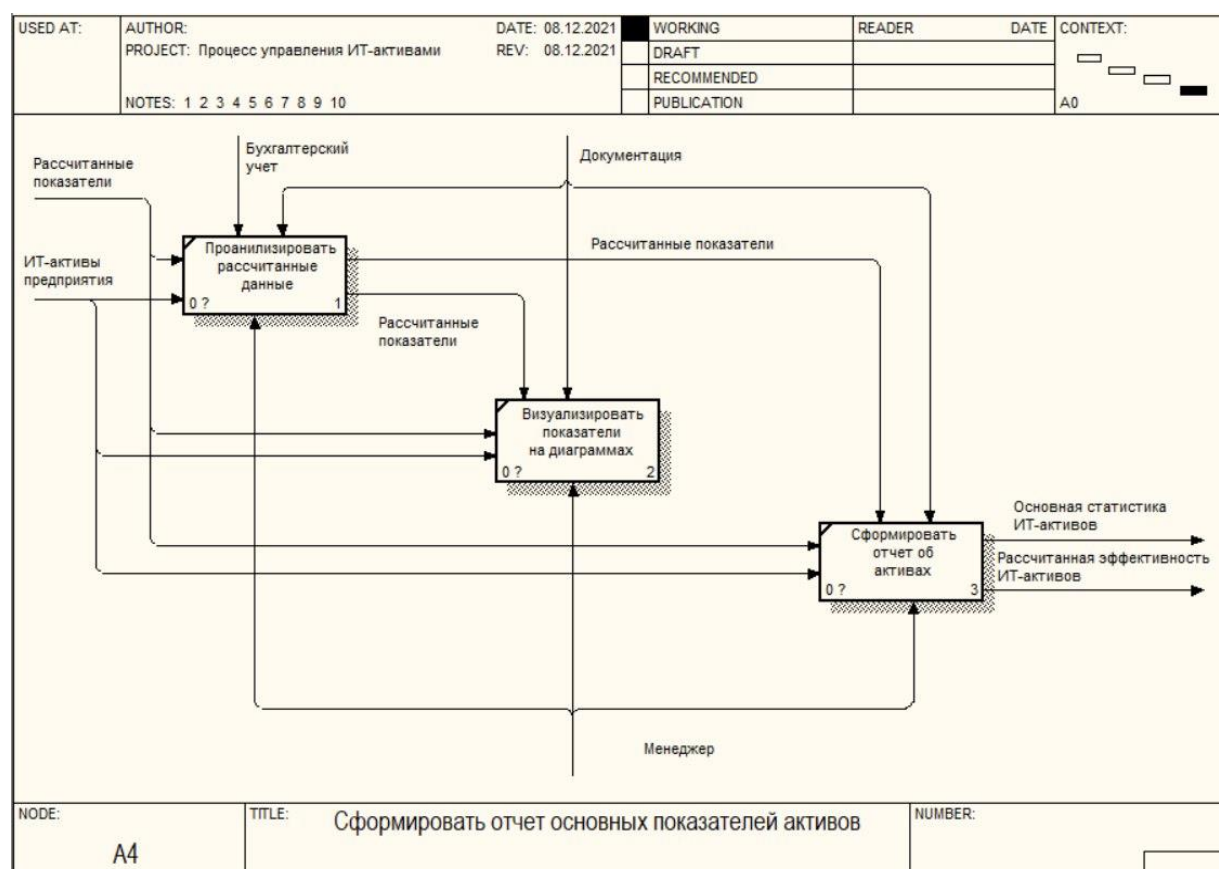


Рисунок 3.5 – Декомпозиция блока «Формирование отчета основных показателей активов»

В результате выполнения описанных процессов структурируется информация о процессе управления ИТ-активами. В дальнейшем это облегчает процесс работы с данными.

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

Информационная система – это коммуникационная и вычислительная система, которая предназначена для сбора, хранения, обработки и передачи информации. Она также снабжает работников различного ранга той информацией, которая необходима им для реализации функций управления.

Информационная модель системы отражает информацию о предметной области, данные о которой должны храниться в проектируемой базе данных. В данном курсовом проекте предметной областью является управление активами предприятия.

Для обеспечения минимальной избыточности и физического объёма данных, а также для более быстрого доступа, модель приведена к 3 нормальной форме.

Для моделирования системы существует 3 этапа проектирования:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

Концептуальным проектированием является анализ предметной области, а также её описание.

Следующим этапом является логическое проектирование. Оно представляет собой описание логической структуры данных посредством системы управления базами данных (СУБД), для которой проектируется БД.

Рассмотрим модель системы (см. рисунок 4.1).

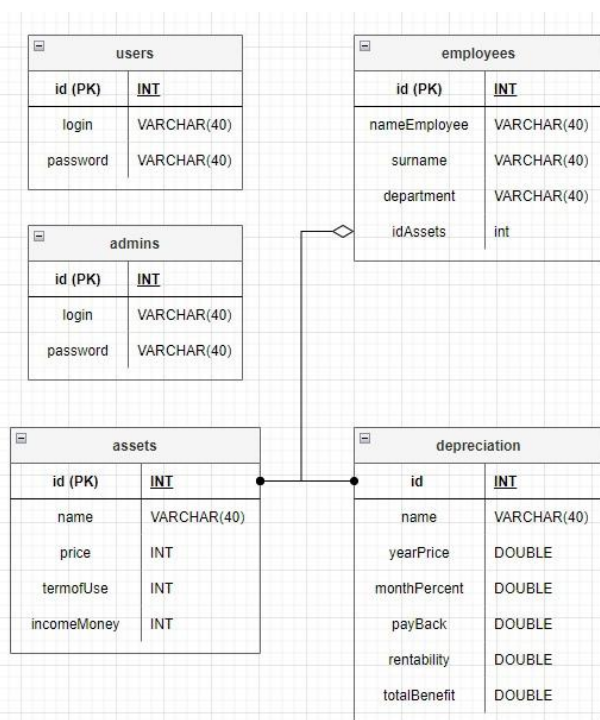


Рисунок 4.1 – Логическая модель системы

Система содержит 5 сущностей: пользователи (user), админы (admins), активы (assets), основные расчеты (depreciation), сотрудники (employees). Сущность – это любой различимый объект, о котором необходимо хранить информацию в базе данных. Каждая сущность содержит атрибуты. Они отображают качества и свойства объекта. У каждой сущности есть первичный ключ, который хранит уникальный номер.

Сущность «Активы» имеет 5 атрибутов: id актива, наименование актива, стоимость актива, срок годности и общая выгода.

Сущность «Администратор» необходима для предоставления возможности входа в систему администратору. Наличие такой сущности позволяет разграничить пользователей системы по типу (администратор или простой пользователь) и тем самым предоставить каждому пользователю необходимый ему функционал. Атрибутами этой сущности являются логин, пароль и уникальный ID для каждого администратора.

Сущность «Пользователь» служит для предоставления возможности входа в систему простому пользователю. Наличие такой сущности позволяет разграничить пользователей системы по типу (администратор или простой пользователь) и тем самым предоставить каждому пользователю необходимый ему функционал. Атрибутами этой сущности являются логин, пароль и уникальный ID для каждого пользователя.

Помимо этого, система позволяет рассчитывать необходимые показатели эффективности активов, данный процесс происходит в автоматическом режиме, т.е. система, на основе загруженных данных производит вычисления и заполняет необходимые данные в базу, после чего с ними можно работать. После того, как информация занесена в систему, на ее основе можно формировать отчеты по имеющимся активам и их основной статистике, а также для более подробного представления, существует возможность визуализировать данные по определенным параметрам. В разделе графиков можно сравнивать активы по некоторым из их показателей.

С учётом обозначенного взаимодействия сущностей смоделируем их взаимодействие в формате IDEF1.X и приведём эту модель к третьей нормальной форме. В результате последовательного приведения получается модель, соответствующая условиям третьей нормальной формы – не ключевой атрибут сущности функционально зависит только от всего первичного ключа и ни от чего другого.

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

При проектировании и реализации программных систем разработчики часто сталкиваются с такими понятиями, как бизнес-логика, бизнес-правила, бизнес-ограничения и т.д. Для того, чтобы раскрыть данные понятия термин «бизнес» можно заменить на понятие предметная область.

Предметная область – это часть реального мира занимающееся деятельностью, которая служит объектом автоматизации.

Бизнес-правило – это положение, определяющее или ограничивающее какие-либо стороны бизнеса.

Бизнес-логика приложения – это описание схем, по которым приложение взаимодействует с пользователем. Например, когда пользователь авторизуется в системе, эти действия обрабатываются на сервере. Бизнес-логика в данном случае отвечает на вопросы: Что должен ввести пользователь?

С точки зрения разрабатываемой системой управления активами на предприятии одним из основных алгоритмов, реализующим бизнес-логику, является работа с рассчитываемыми данными активов. Расчет включает в себя следующие позиции (см.рисунок 5.1):

- расчет окупаемости;
- расчет рентабельности;
- расчет месячного износа;
- расчет общей выгоды.

После успешного входа в систему и перехода пользователя на вкладку «Добавление данных» пользователю предоставляются меню, где можно выбрать необходимый блок работы с данными, это могут быть как данные сотрудников, основная информация об активах, а также расчет их эффективности на предприятии. При этом информация заносится в базу данных после ввода соответствующих полей и сохранения их в системе. Пользователю необходимо выбрать интересующий его блок данных, после чего нажать на кнопку и перейти в соответствующий интерфейс добавления.

После добавления всех необходимых данных для расчета эффективности актива, необходимо нажать на кнопку «Добавить». После этого осуществляется запрос к базе данных на добавление информации о активе. После сохранения данных, пользователь может либо произвести добавление об ещё одном активе или же перейти в другое меню для дальнейшей работы.

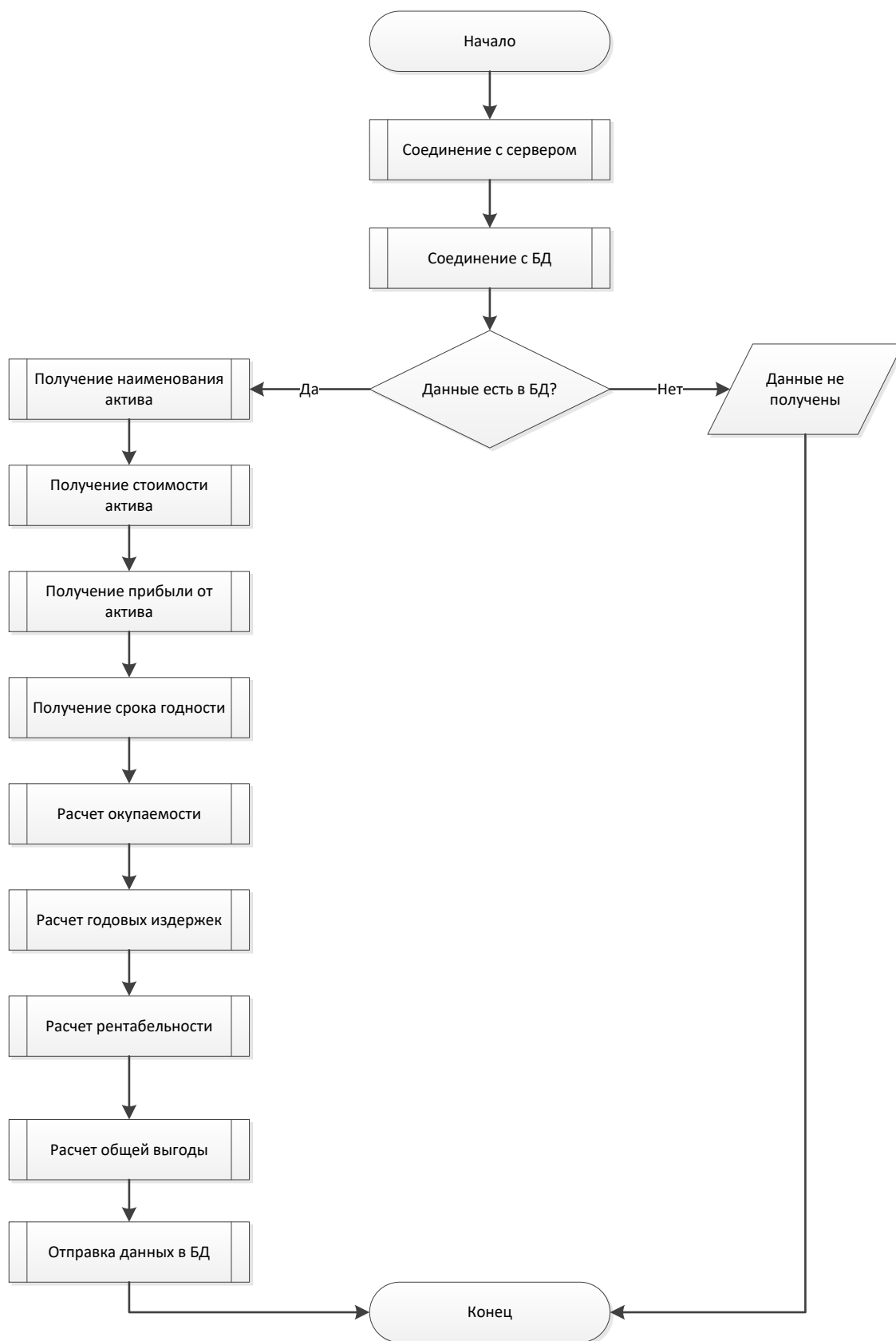


Рисунок 5.1 – Алгоритм расчета показателей эффективности актива

Ещё одним важным этапом курсовой работы является авторизация пользователей (см. рисунок 5.2). Приложение получает введённые данные, проверяет их и производит авторизацию и выдачу соответствующих прав, если данных пользователя нет, их можно добавить в систему путем регистрации нового аккаунта, после чего можно произвести вход в систему.

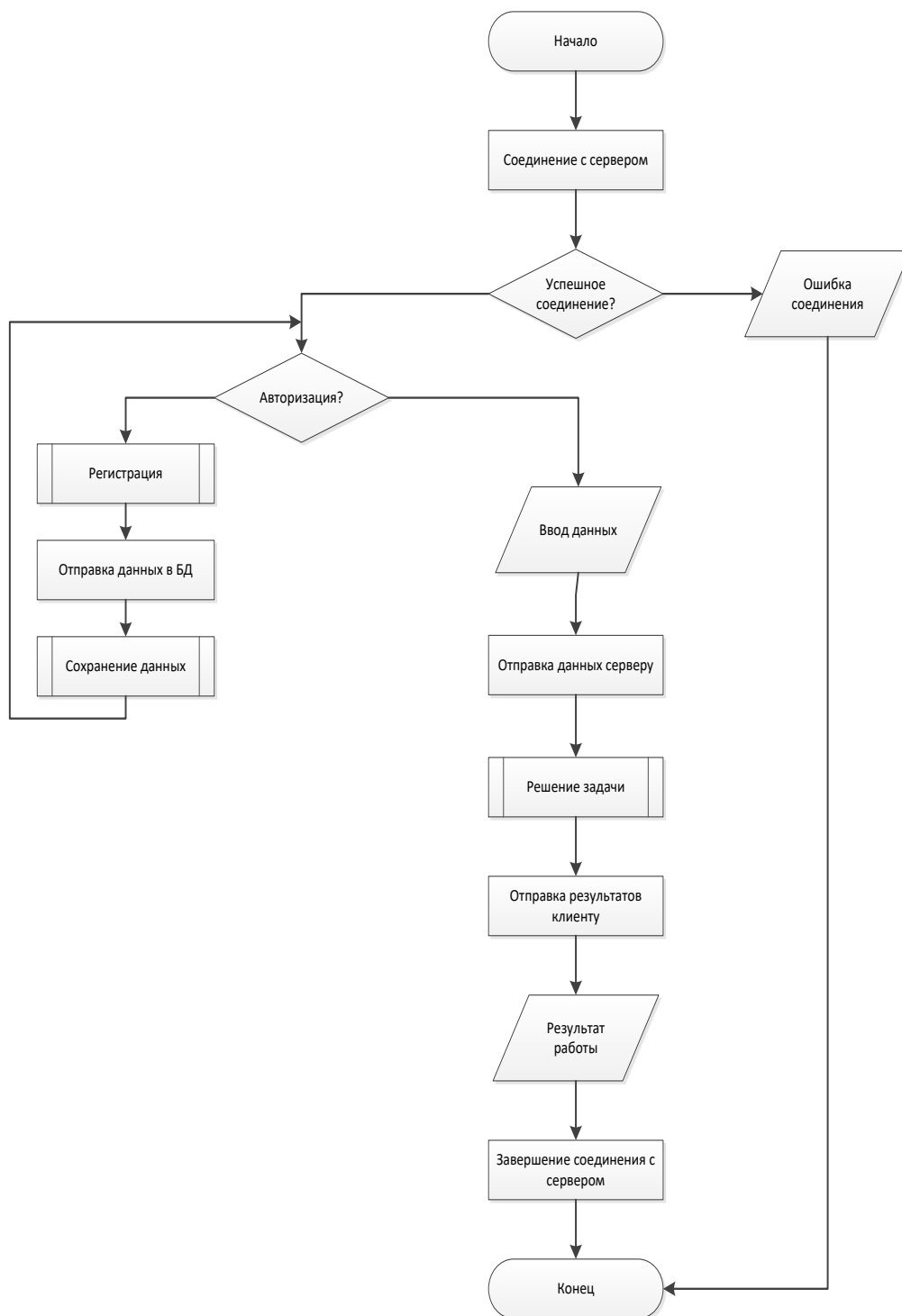


Рисунок 5.2 – Алгоритм авторизации в системе

6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

UML (англ. Unified Modeling Language) – унифицированный язык моделирования. Он состоит из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем, а также, к примеру, для бизнес-моделирования.

Одна из задач UML – служить средством коммуникации внутри команды и при общении с заказчиком.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы. На общем уровне графическая нотация UML содержит 4 основных типа элементов:

- фигуры;
- линии;
- значки;
- надписи.

Для создания диаграмм можно использовать различные сервисы, такие как diagrams.net, google drawings, microsoft visio и т.д.

Диаграммы UML подразделяют на два типа — это структурные диаграммы и диаграммы поведения. Структурные диаграммы показывают статическую структуру системы и ее частей на разных уровнях абстракции и реализации, а также их взаимосвязь [14]. Существует семь типов структурных диаграмм. Среди них можно выделить следующие диаграммы:

- диаграмма классов;
- диаграмма развертывания;
- диаграмма компонентов.

Диаграммы поведения показывают динамическое поведение объектов в системе, которое можно описать, как серию изменений в системе с течением времени. К диаграммам поведения относятся:

- диаграмма прецедентов;
- диаграмма состояний;
- диаграмма последовательности.

Перечисленные выше диаграммы были построены для разработанной системы.

6.1 Диаграмма классов

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними [13].

Диаграмма классов дает наиболее полное и развернутое представление о связях в программном коде, функциональности и информации об отдельных классах.

Выделяют 3 наиболее важных типа связей на диаграмме классов: ассоциация, агрегация и наследование. Ассоциация показывает отношения между экземплярами классов. Агрегация показывает связь целого с частью (например, когда один класс является частью другого).

Диаграмма классов проектируемой системы представлена на рисунке 6.1. на диаграмме отображены связи классами. **ClientHandler** – класс, который реализуют взаимодействия пользовательских запросов с базой данных. В классе **DataObject** представлены основные элементы модели, использующиеся в программе.

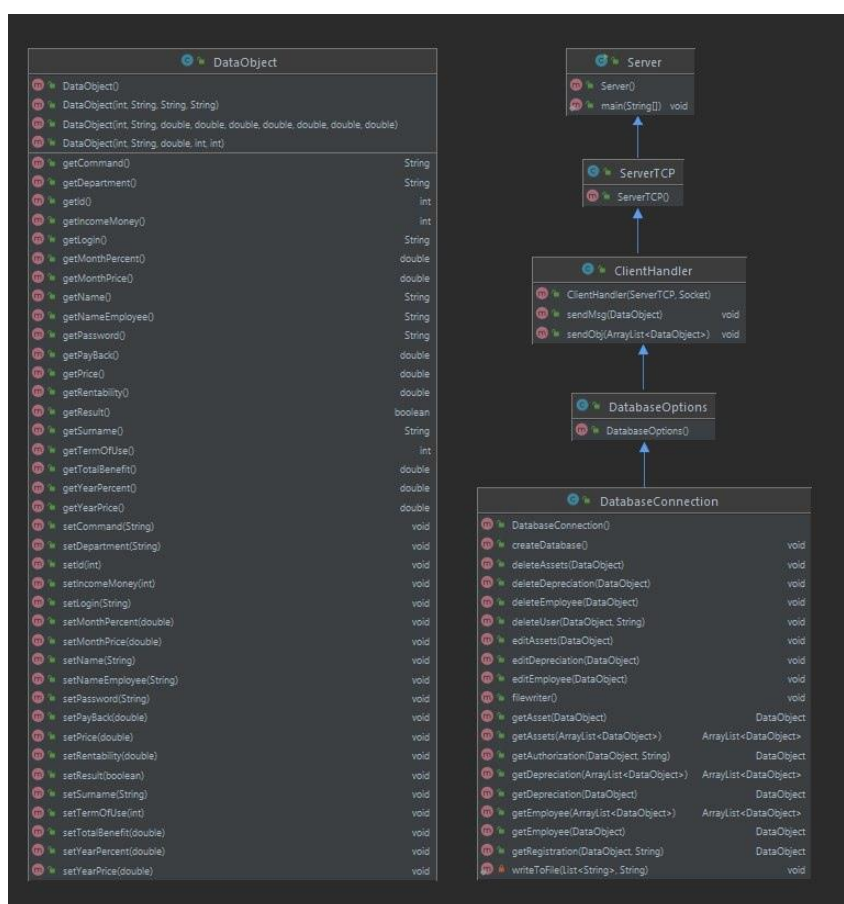


Рисунок 6.1 – Диаграмма классов

При проектировании и реализации программных систем разработчики часто сталкиваются с такими понятиями, как бизнес-логика, бизнес-правила, бизнес-ограничения и т.д. Для того, чтобы раскрыть данные понятия термин «бизнес» можно заменить на понятие предметная область.

6.2 Диаграмма компонентов

На языке унифицированного моделирования диаграмма компонентов показывает, как компоненты соединяются вместе для формирования более крупных компонентов или программных систем.

Она иллюстрирует архитектуры компонентов программного обеспечения и зависимости между ними.

Другими словами, диаграмма компонентов дает упрощенное представление о сложной системе, разбивая ее на более мелкие компоненты. Каждый из элементов показан в прямоугольной рамке с названием, написанным внутри. Соединители определяют отношения и зависимости между различными компонентами [13].

Среди основных целей диаграммы компонентов можно выделить:

- визуализация общей структуры исходного кода программы;
- обеспечение многократного использования фрагментов кода;
- представление концептуальной и физической базы данных.

Диаграмма компонентов представлена на рисунке 6.2. Она позволяет определить состав компонентов программы, а также установить зависимость между ними.

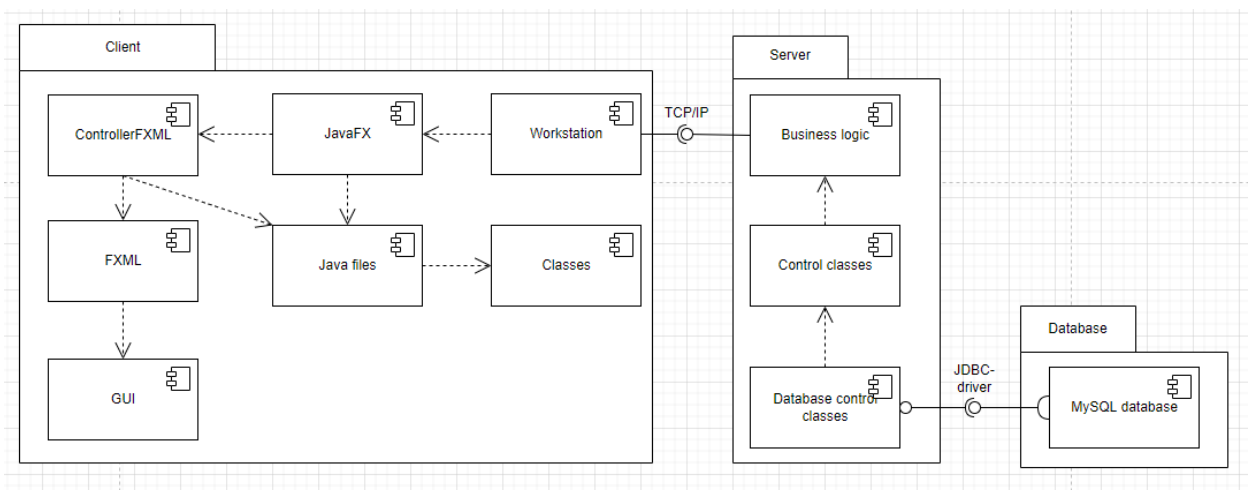


Рисунок 6.2 – Диаграмма компонентов

6.3 Диаграмма развертывания

Диаграмма развертывания помогает моделировать физический аспект объектно-ориентированной программной системы. Это структурная схема, которая показывает архитектуру системы, как развертывание (дистрибуции) программных артефактов. То есть на этой диаграмме показаны аппаратные (узлы) и программные (артефакты) компоненты и их взаимосвязи. Она предлагает наглядное представление о том, где именно развернут каждый программный компонент.

Артефакты представляют собой конкретные элементы в физическом мире, которые являются результатом процесса разработки.

Диаграмма моделирует конфигурацию времени выполнения в статическом представлении и визуализирует распределение артефактов в приложении.

В большинстве случаев это включает в себя моделирование конфигураций оборудования вместе с компонентами программного обеспечения, на которых они размещены.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единственной для системы в целом, поскольку должна отражать все особенности ее реализации. Разработка диаграммы развертывания, как правило, является последним этапом спецификации модели. Диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками.

Кроме того, диаграмма развертывания позволяет решить такую задачу, как обеспечение безопасности системы.

Разработанная диаграмма развертывания представлена на рисунке 6.3. Цели разработки данной диаграммы можно сформулировать так:

- распределение компонентов системы по ее физическим узлам;
- отображение физических связей между узлами системы на этапе исполнения;
- выявление узких мест системы и реконфигурация ее топологии с целью достижения требуемой производительности.

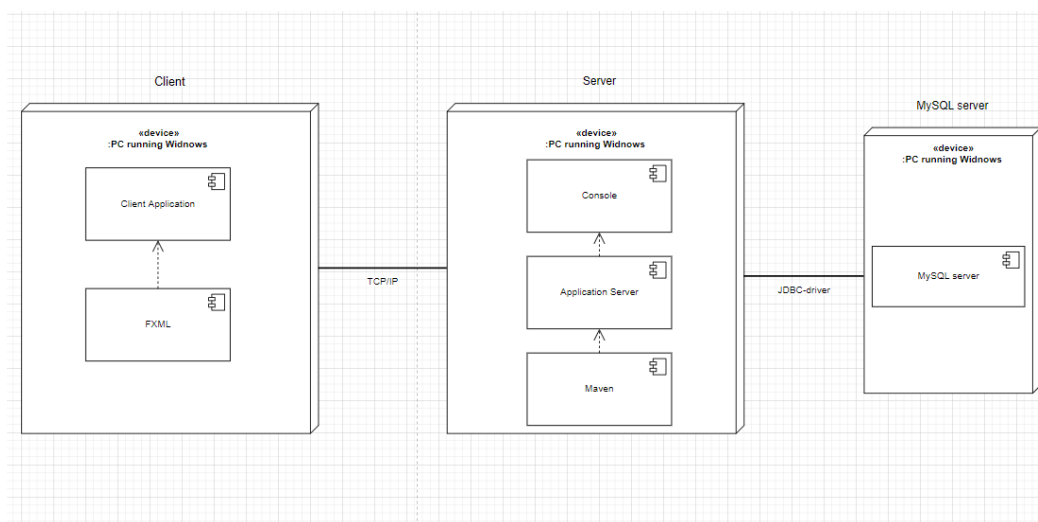


Рисунок 6.3 – Диаграмма развертывания

От правильности построения данной диаграммы зависит и производительность системы, т.к. позволяет рационально распределить компоненты по узлам системы.

6.4 Диаграмма состояний

Диаграмма состояний относится к поведенческому виду. Она показывает, как объект переходит из одного состояния в другое.

Все объекты в системе характеризуются поведением и состоянием, в котором они находятся.

Состояние (state) - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

Диаграммы состояний отображают разрешенные состояния и переходы, а также события, которые влияют на эти переходы. Она помогает визуализировать весь жизненный цикл объектов и, таким образом, помогает лучше понять системы, основанные на состояниях.

Очевидно, что диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и, как мы увидим далее, диаграммы деятельности).

Диаграмма состояний полезна при моделировании жизненного цикла объекта.

От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

На рисунке 6.4 представлена диаграмма состояний приложения на которой можно наблюдать основные переходы.

Изначально пользователю необходимо произвести авторизацию, после чего в соответствии с предоставленными данными, ему будет доступны определенные функциональные возможности. Обычные пользователи могут просматривать, добавлять информацию об активах, рассчитывать их эффективность. Помимо этого всем пользователям доступен функционал формирования отчета для дальнейшего взаимодействия с данными. Любой пользователь на основе полученных данных может построить график для наиболее наглядного представления всей статистики.

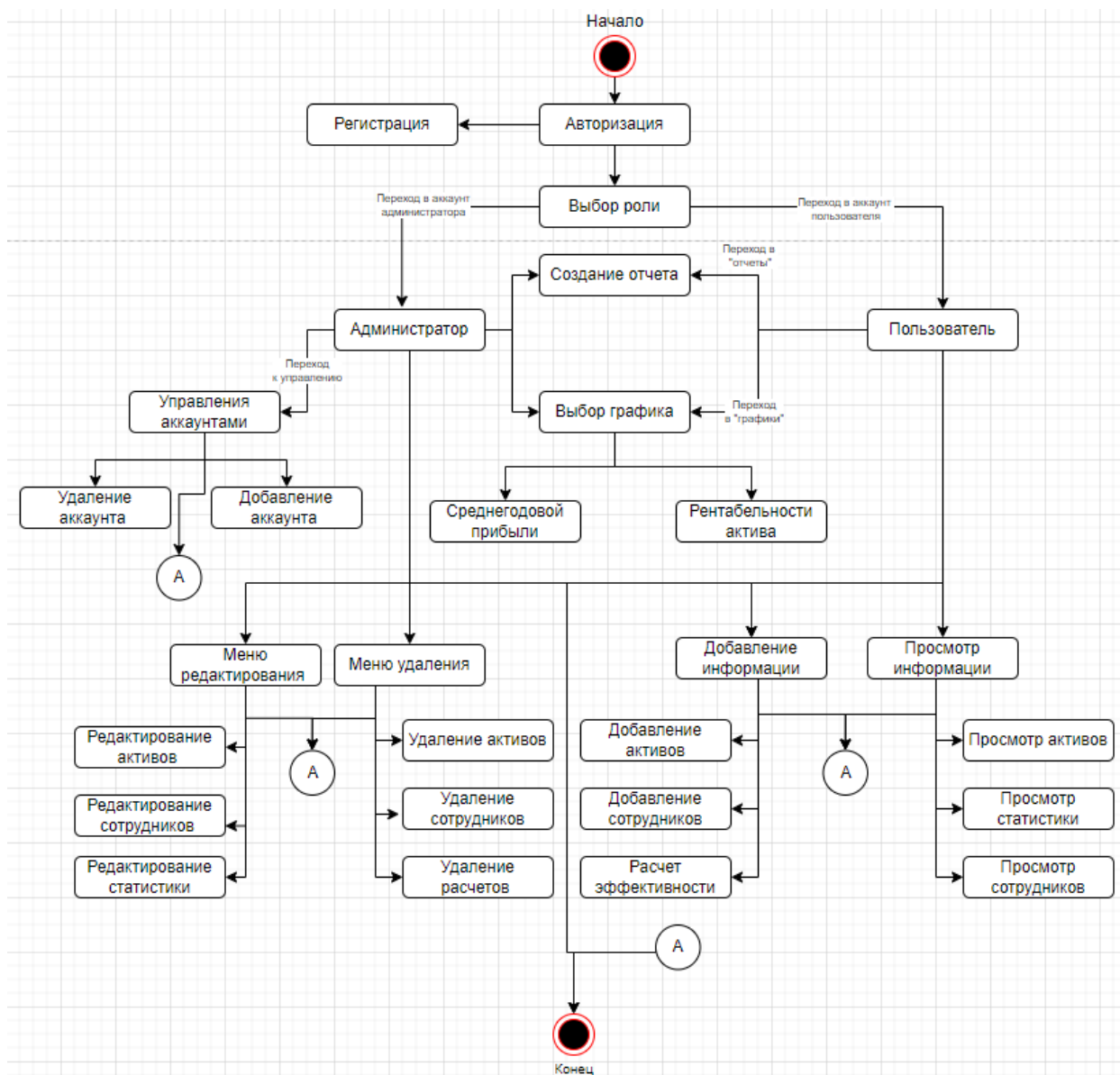


Рисунок 6.4 – Диаграмма состояний сотрудника при добавлении

Скругленные прямоугольники представляют собой состояния, через которые проходит объект в течение своего жизненного цикла. Стрелками обозначаются переходы между состояниями, которые вызваны выполнением некоторых действий.

6.5 Диаграмма последовательности

Диаграмма последовательности – это поведенческий тип UML диаграмм, использующийся для описания логики сценариев использования.

Диаграмма последовательности моделирует взаимодействие объектов на основе временной последовательности. Она показывает, как одни объекты взаимодействуют с другими в конкретном прецеденте.

Данный вид диаграммы содержит объекты, которые взаимодействуют в рамках сценария, сообщения, передаваемые другим объектам, а также получаемые ответы. У каждого объекта есть определенная линия жизни, в течение которой он способен функционировать и общаться с другими объектами [16].

На рисунке 6.5 представлена диаграмма последовательности, которая демонстрирует работу программы.

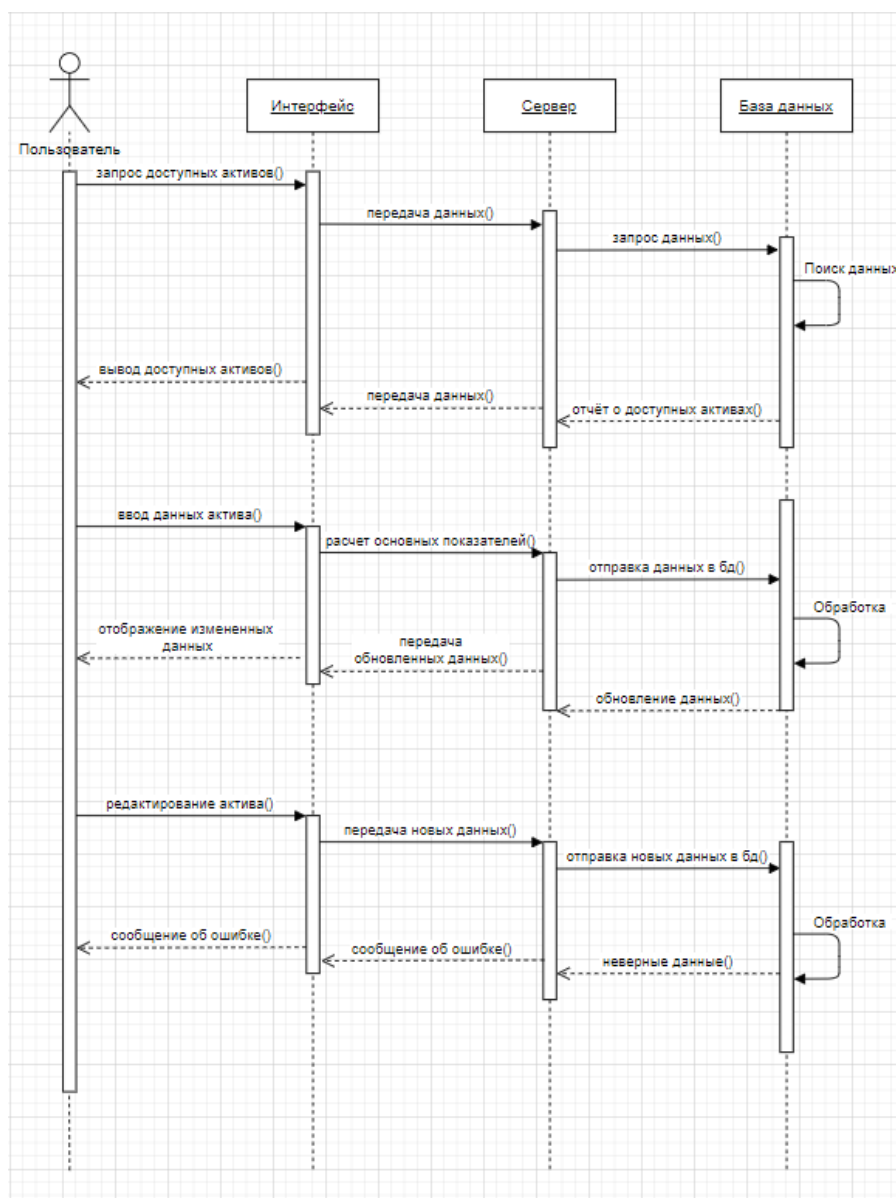


Рисунок 6.5 – Диаграмма последовательности просмотра данных сотрудника

Изначально, пользователь запрашивает необходимые ему данные по активам, после чего эта информация из интерфейса передается серверу, где происходит их обработка и дальнейшее направление в базу данных, где запрос будет принят и начнется поиск данных, после чего пользователю

будет отправлен отчет о доступных активах, который передается в сервер, а после и в интерфейс для возможно работы клиента с полученными данными.

6.6 Диаграмма вариантов использования

Диаграмма последовательности – это поведенческий тип UML диаграмм, использующийся для описания логики сценариев использования.

Диаграмма вариантов использования - это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки. Создание диаграммы вариантов использования имеет следующие цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования использует 2 основных элемента.

Участник, или действующее лицо (англ. actor)— множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек, роль человека в системе или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецедент, или вариант использования (англ. use case) — описание отдельного аспекта поведения системы с точки зрения пользователя. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.

Между вариантами использования и действующими лицами на диаграмме вариантов использования устанавливаются следующие 4 типа отношений:

- отношение ассоциации;
- отношение обобщения;
- отношение включения;
- отношение расширения.

Ассоциация – это связь между вариантом использования и действующим лицом.

Обобщение означает, что один вариант использования или актер может быть обобщен от другого. Тогда первый вариант использования будет дочерним, а второй – родительским. Следует отметить, что дочерний вариант использования наследует от родительского все свойства, но при этом может иметь и свои свойства.

Отношение включения (англ. include) означает, что поведение одного варианта использования является обязательным компонентом поведения другого.

Отношение расширения (англ. extend) применяется тогда, когда свойства одного варианта использования могут быть дополнены свойствами другого.

На рисунке 6.6 представлена диаграмма вариантов использования для администратора и пользователя. Они изображены на ней в виде актеров. Стоит отметить, что актер на диаграмме вариантов использования – это собирательный образ и не означает конкретно одного человека. Например, актер «Администратор» не обозначает одного конкретного администратора, а всех пользователей системы, имеющих роль «Администратор». Таких пользователей может быть сколь угодно много.

В виде овалов на диаграмме вариантов использования приведены варианты использования системы, которые связаны с актерами отношением ассоциаций. При этом некоторые варианты использования связаны друг с другом отношениями включения или обобщения.

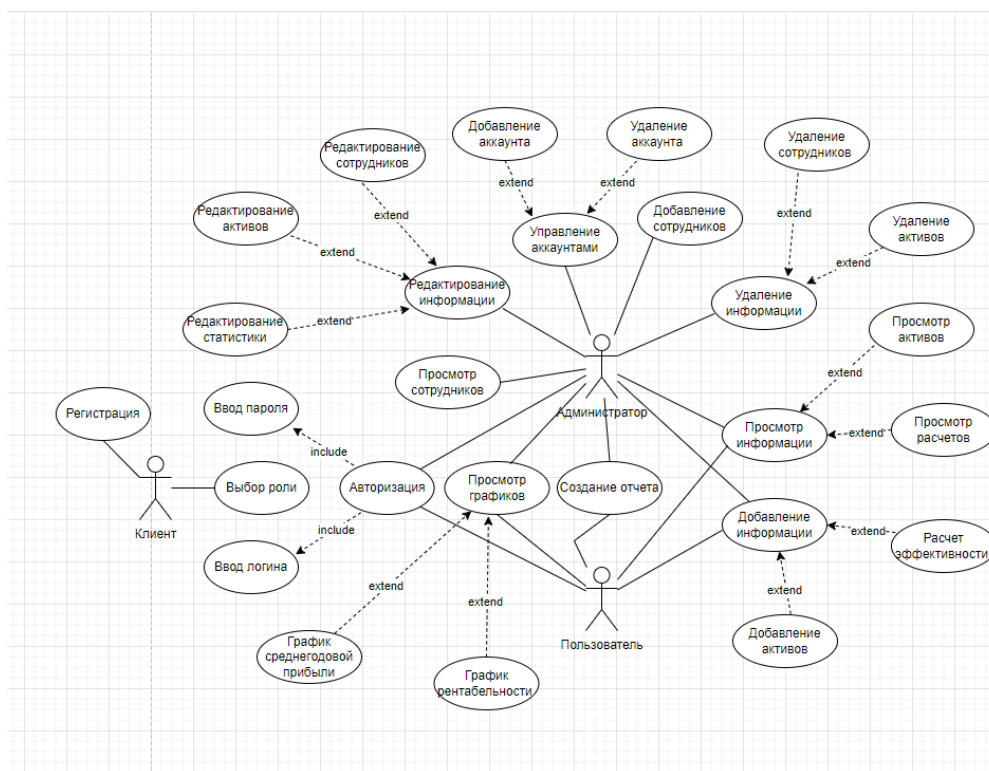


Рисунок 6.6 – Диаграмма вариантов использования

Благодаря представленной информации на диаграмме, мы можем проследить существующие варианты использования приложения определенными пользователями.

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Перед началом работы приложения необходимо установить соединение с сервером, для этого необходимо изначально запустить серверную часть. При успешно установлении соединения, запускается клиентская часть, в которой можно выполнять основной функционал. Изначально пользователь попадает в меню авторизации в системе, где ему необходимо выбрать в качестве какой роли в дальнейшем он будет производить работу с приложением. Можно выбрать как администратора, так и пользователя (см.рисунок 7.1)

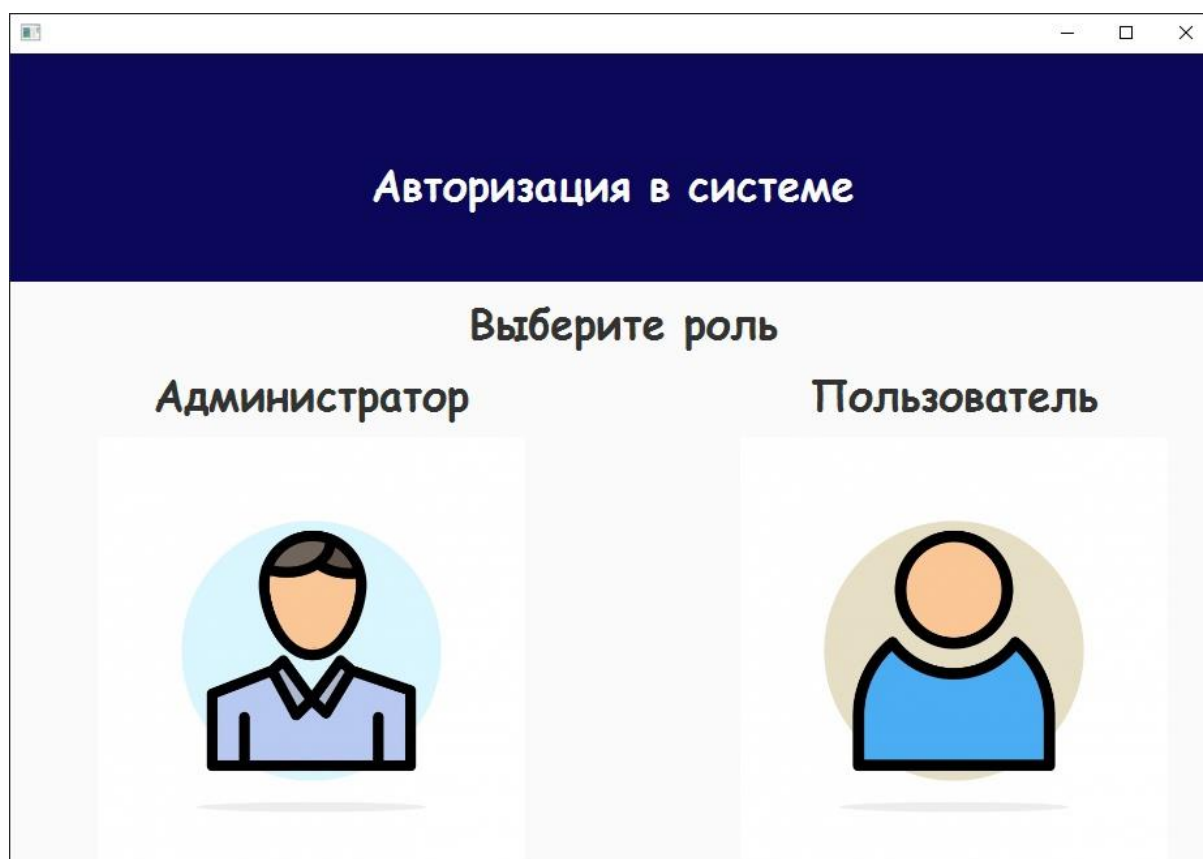


Рисунок 7.1 – Выбор роли при входе

Производим вход в качестве пользователя. После чего у нас появляется выбор регистрации нового пользователя или войти в уже существующий аккаунт, для этого необходимо ввести свой логин и пароль. Если пользователь вводит некорректные данные, вход не будет выполнен до того момента, пока данные не совпадут с существующим аккаунтом. При этом, при некорректном вводе производится анимация (см.рисунок 7.2)

The image shows a web application window titled "Авторизация пользователя" (User Authorization). The window has a dark blue header with the title in white. Below the header, on a light gray background, are five buttons arranged vertically. The first two buttons are for entering login and password, the third is for authorization, and the last two are for registration and returning to the previous screen.

Введите логин

Введите пароль

Авторизироваться

Зарегистрироваться

Вернуться назад

Рисунок 7.2 – Авторизация пользователя

Если пользователь ввел корректные данные, после чего нажал на кнопку авторизации, произойдет вход в систему с использованием пользовательских прав. При этом пользователю открывается определенный функционал для работы с данными. В основном меню, представленном пользователю, присутствует раздел просмотра, добавления информации, а также вывод различного вида графиков для визуализации данных. Помимо этого пользователь может сформировать отчет по существующим параметрам активов для дальнейшей работы с ними (см.рисунок 7.3).

Графическое представление данных позволяет наиболее наглядно визуализировать систему для дальнейшего анализа представленных данных, на графиках четко прослеживаются показатели каждого из активов, поэтому их легко можно сравнить между собой по определенным параметрам.

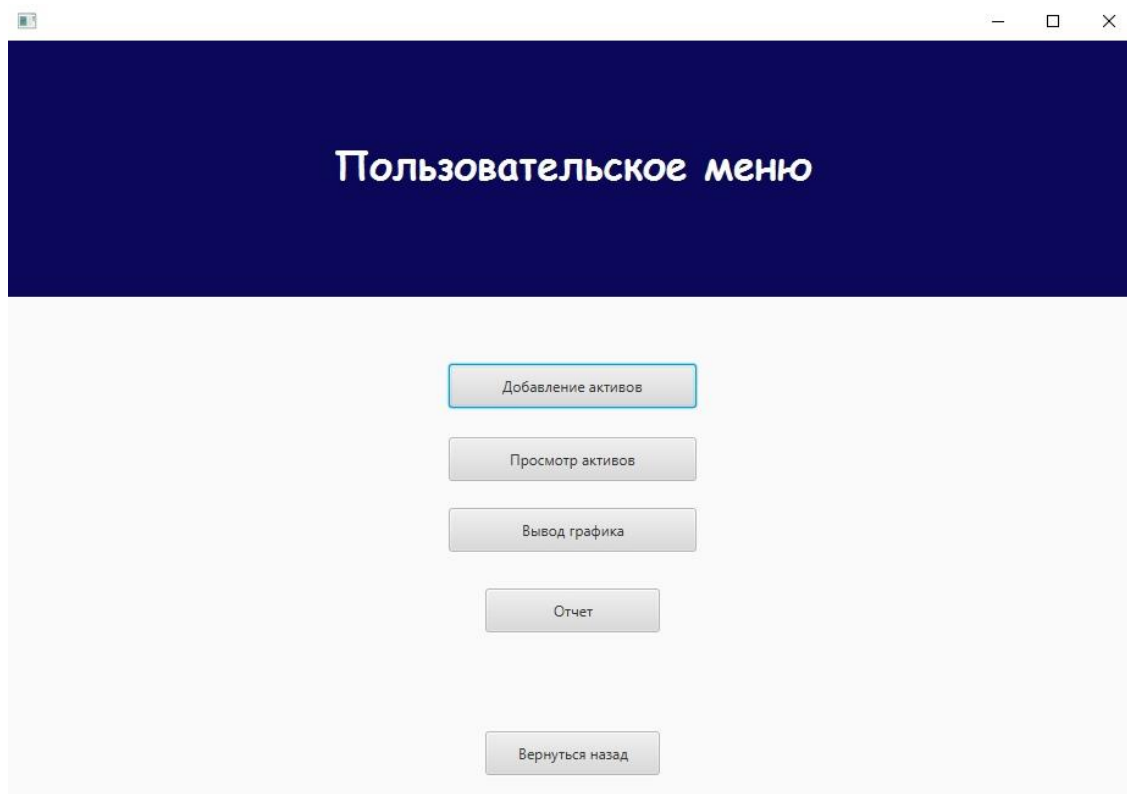


Рисунок 7.3 – Пользовательское меню

Пользователь может просматривать различные данные, представленные в системе (см.рисунок 7.4)

id	Название	Годовые издержки	Месячный износ %	Окупаемость	Рентабельность	Выгода
43	Принтер №1	210.0	0.833	0.7	143.0	27900.0
44	Компьютер №3	286.667	0.556	0.43	233.0	145685.0
45	Компьютер №4	300.0	0.926	0.303	330.0	77391.0
46	Лицензия JK	1800.0	8.333	1800.0	0.0	-1799.0
47	GeForce RTX 3090	799.9	0.833	5.333	19.0	7001.0
48	AMD Radeon RX 6900 XT	707.071	0.595	9.909	10.0	4087.0



Рисунок 7.4 – Просмотр данных

Если пользователь осуществляет вход в качестве администратора, для него открывается следующее основное меню, представленное на рисунке 7.5, функционал которого расширен по сравнению с пользовательскими возможностями.

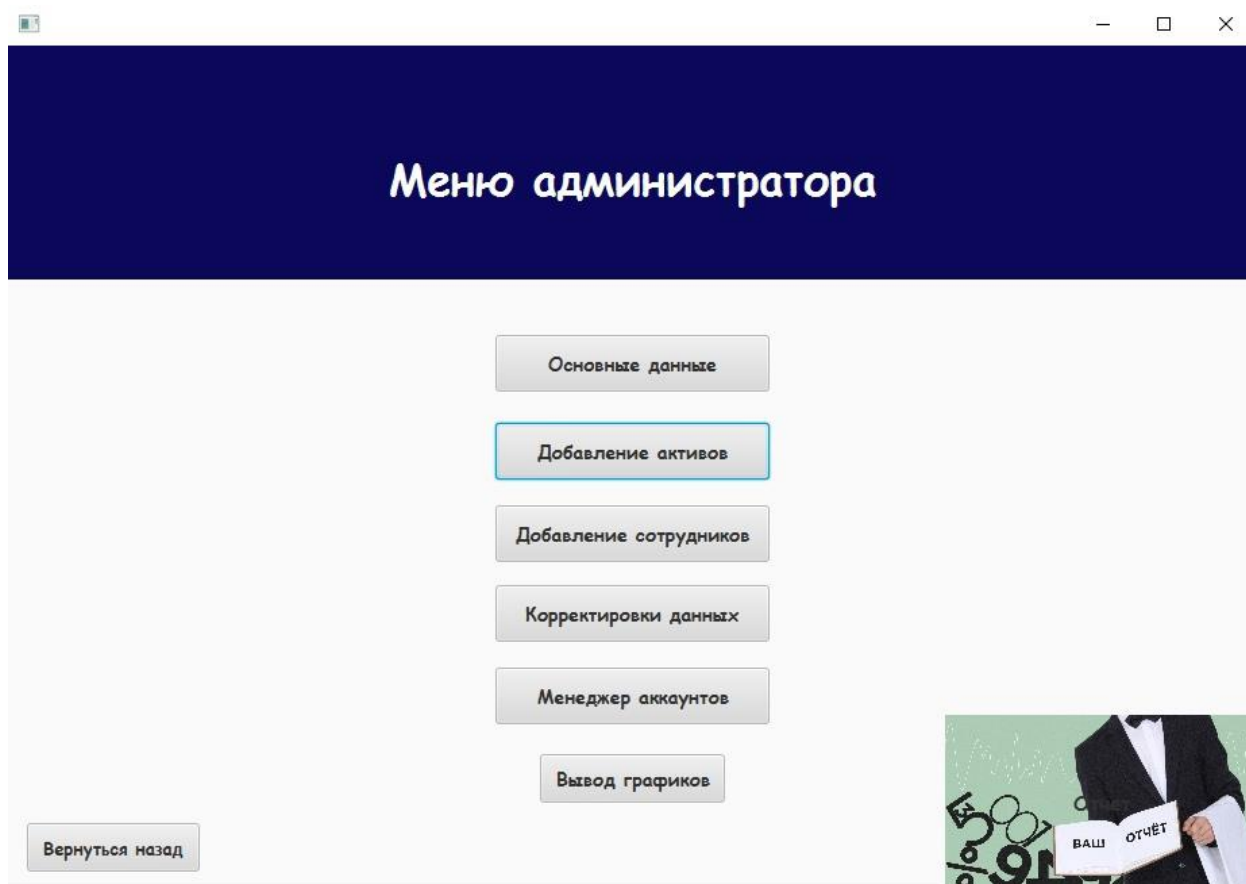


Рисунок 7.4 – Меню администратора

В данном меню представлен перечень функциональных возможностей администратора. Прежде всего, администратор может добавлять данные о новых активах, после чего производить расчеты эффективности и другие показатели, связанные с активами. Если данные об активах уже добавлены в систему, но при этом необходимо произвести определенные манипуляции и обновить данные, администратор переходит в раздел корректирования данных, где может осуществлять изменение необходимых данных. Помимо добавления информации об основных активах, администратор может обновлять сотрудников предприятия, а также в менеджере аккаунтов, настраивать пользовательские данные для входа в систему.

Меню просмотра всех данных в роли администратора представлено на рисунке 7.5, здесь можно просмотреть информацию об активах, помимо этого все основные расчеты эффективности, а также данные сотрудников, которые работают на предприятии.

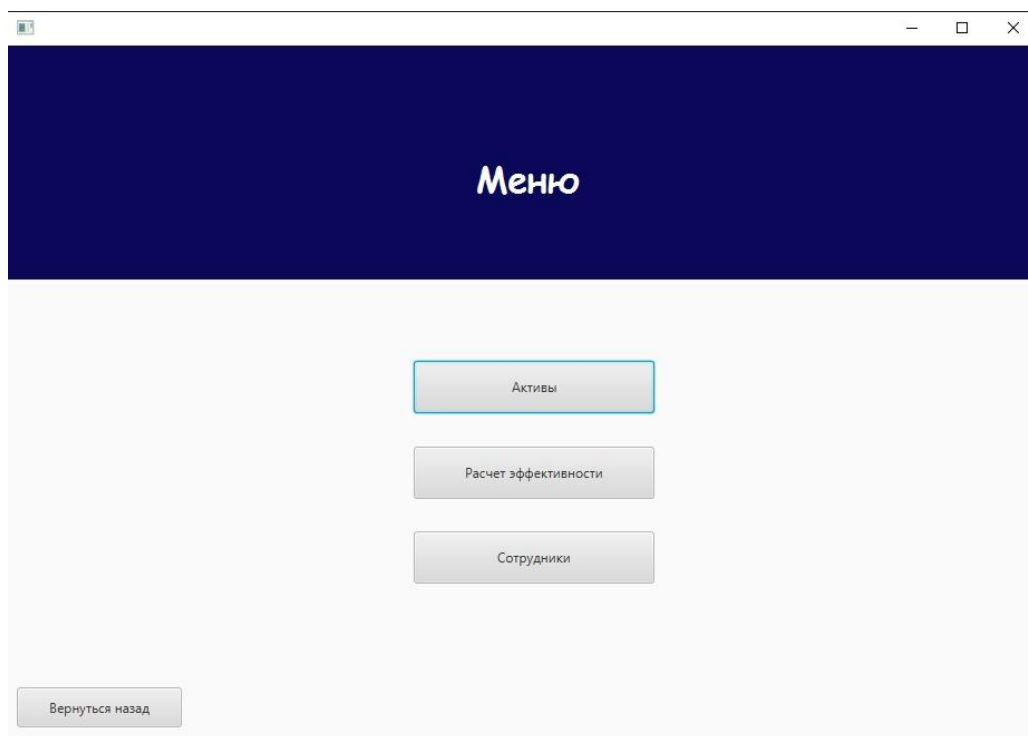


Рисунок 7.5 – Меню просмотра администратора

В аккаунте обычного пользователя и администратора представлен раздел для наиболее эффективной работы с данными. Раздел просмотра графиков изображен на рисунке 7.6, где пользователю необходимо выбрать интересующий его график.

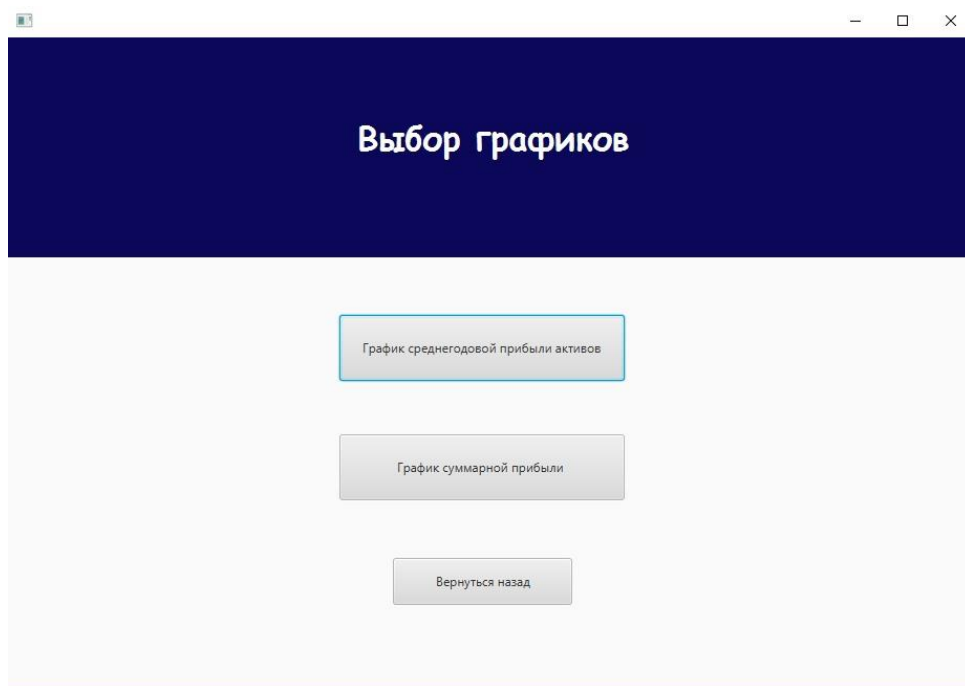


Рисунок 7.6 – Меню выбора графиков

График суммарной прибыли представлен на рисунке 7.7

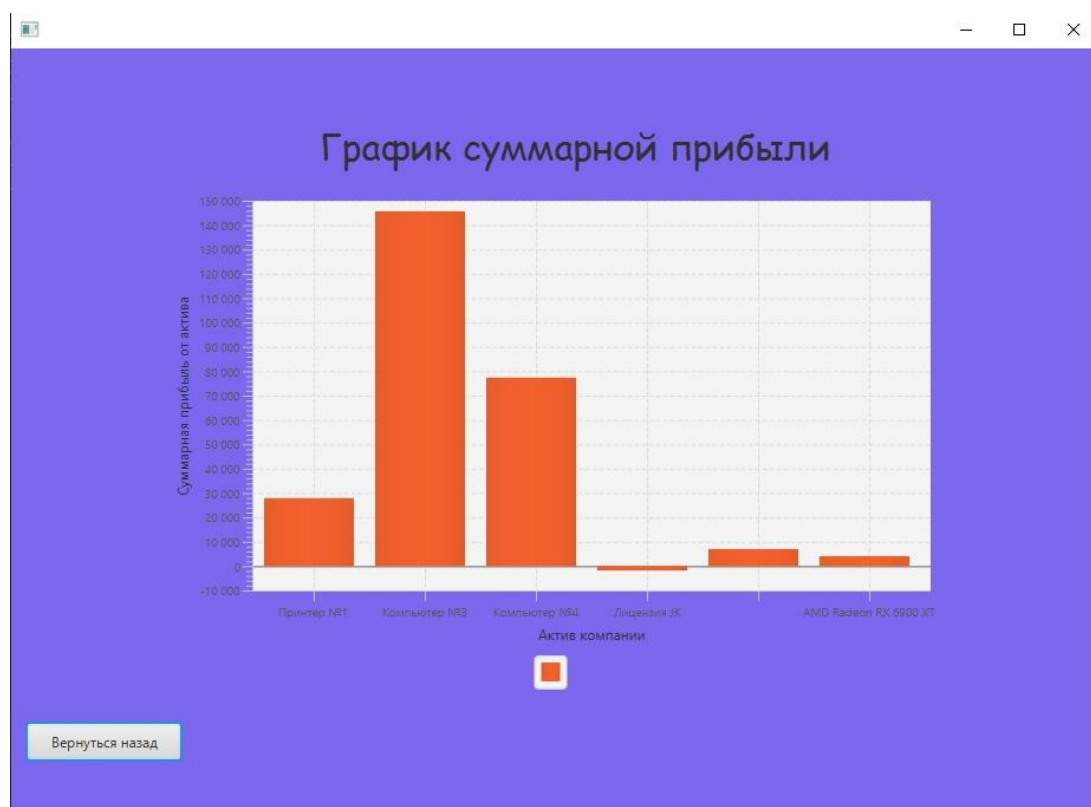


Рисунок 7.7 – График суммарной прибыли

График среднегодовой прибыли активов представлен на рисунке 7.8

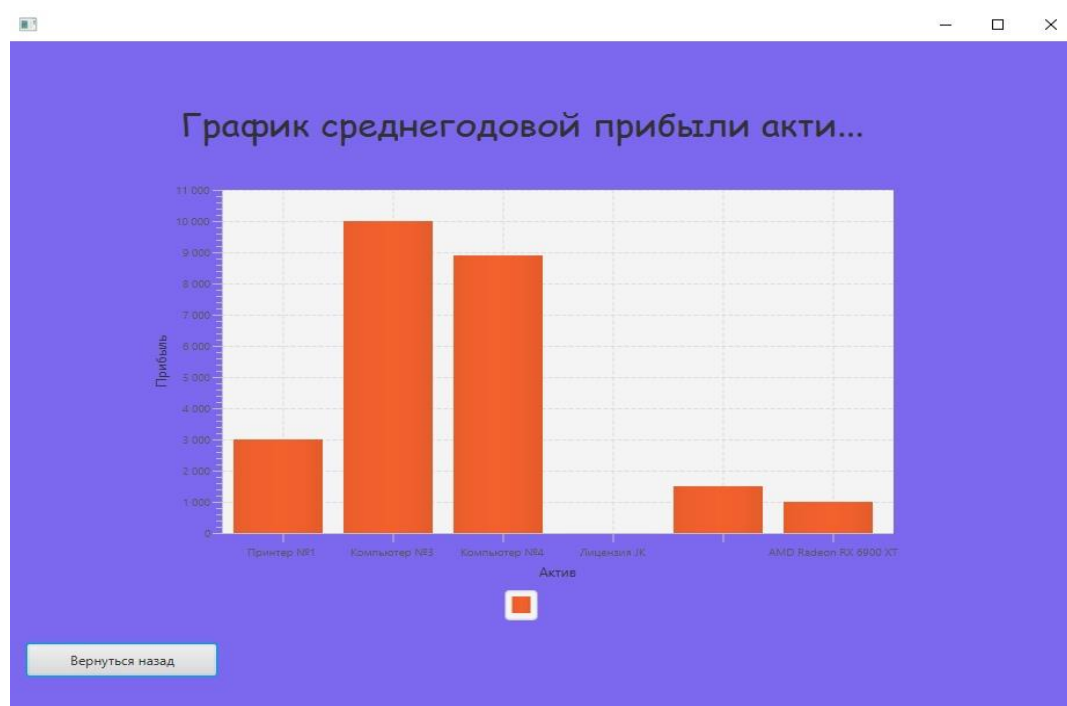
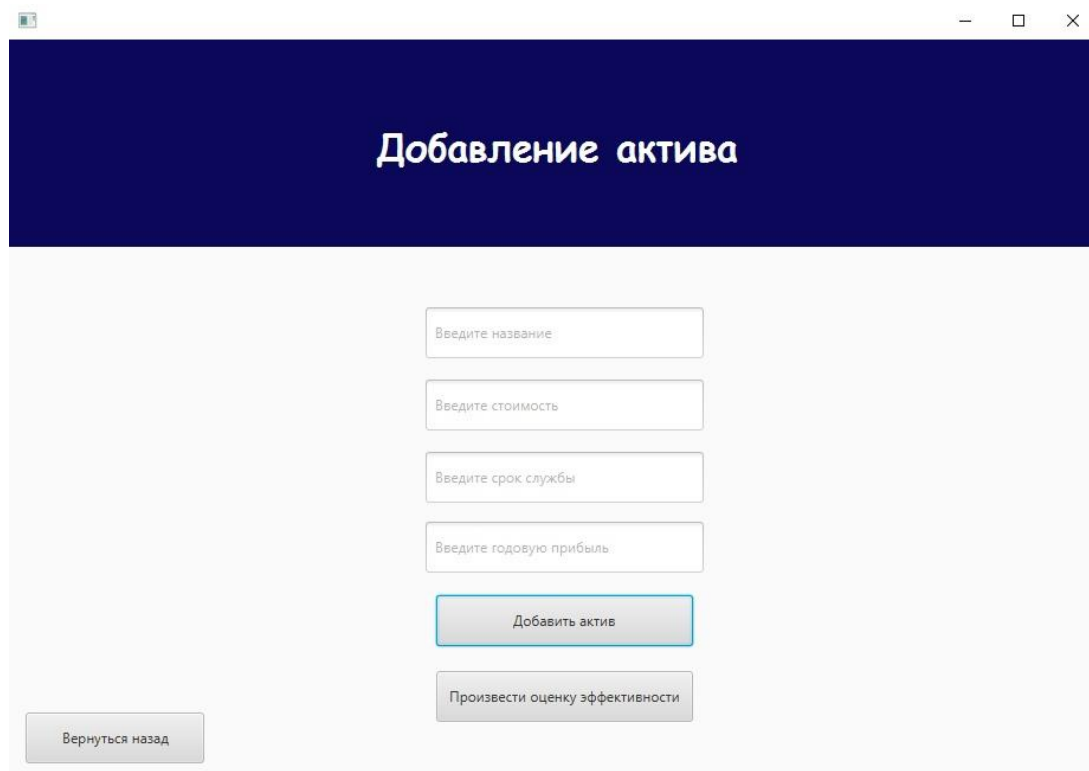


Рисунок 7.8 – График среднегодовой прибыли

Меню добавления активов представлено на рисунке 7.9. Перед пользователем открывается интерфейс с определенными полями, которые ему необходимо заполнить для сохранения информации об активе и последующей работы с этими данными.



Добавление актива

Введите название

Введите стоимость

Введите срок службы

Введите годовую прибыль

Добавить актив

Произвести оценку эффективности

Вернуться назад

Рисунок 7.9 – Меню добавления актива

Когда необходимо добавить информацию об активах в систему, пользователь переходит в раздел добавления, где изначально ему необходимо ввести основные данные об активах, изначально вводится название актива, после чего указывается стоимость и срок службы актива, а также среднегодовую прибыль, после чего актив добавляется в систему, а на основе введенных данных, система автоматически проводит анализ и расчеты эффективности, которые в дальнейшем визуализируются в графиках и также хранятся в системе.

Раздел менеджера аккаунтов представлен на рисунке 7.10. В нем администратор может производить работу с данными пользователей, которые зарегистрированы в системе.

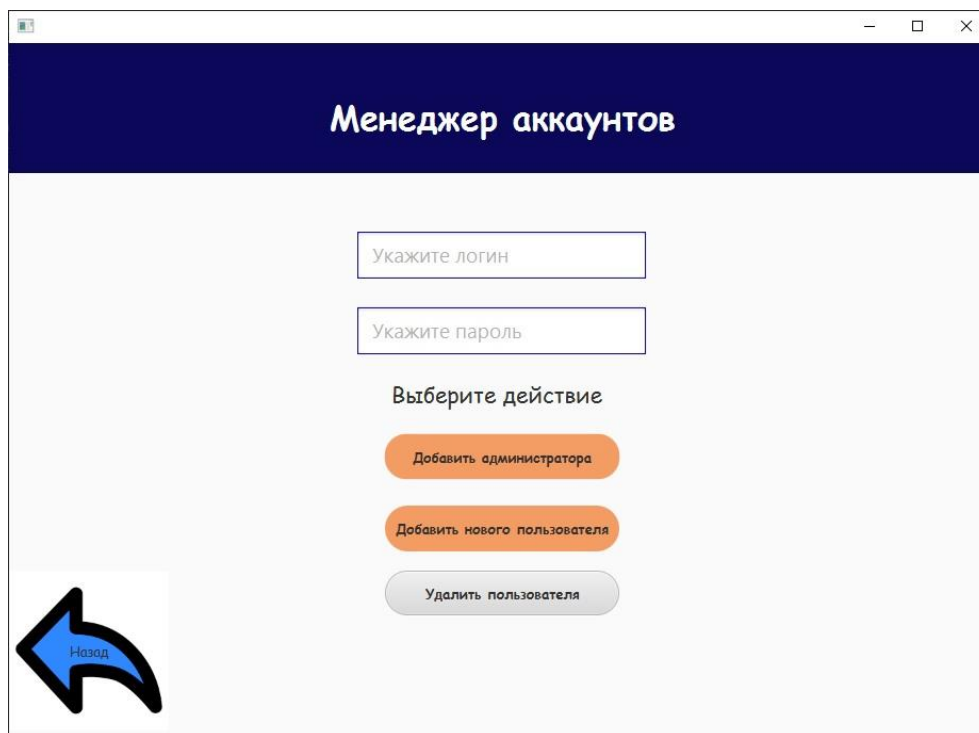


Рисунок 7.10 – Меню менеджера аккаунтов

В данном разделе, администратор производит работу с основными данными аккаунтов, которые могут взаимодействовать с системой. Можно как добавлять аккаунты, которые в дальнейшем смогут произвести вход в систему, а также деактивировать аккаунты пользователей для того, чтобы они не смогли произвести вход в систему.

Это необходимо для того, чтобы каждый сотрудник, который производит работу с определенными данными активов, имел для входа свои идентификационные данные. В случае, если сотрудник покинет организацию, благодаря данному функционалу, администратор просто деактивирует аккаунт пользователя, после чего тот больше не сможет взаимодействовать с информацией, которая хранится в системе.

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ИТ-АКТИВАМИ

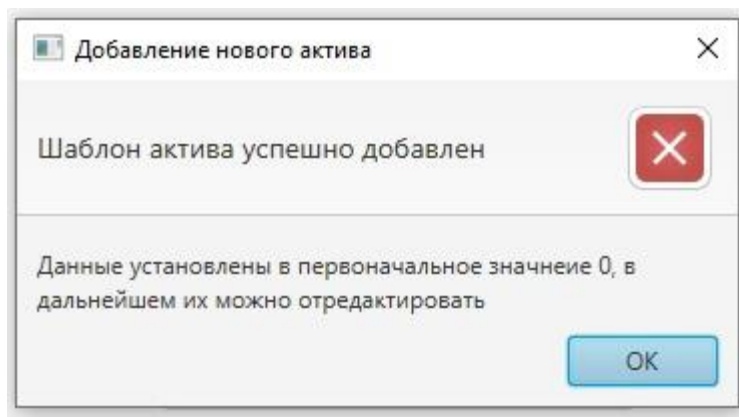


Рисунок 8.1 – Информация об добавлении актива

Следующий раздел администратора необходим для редактирования основных данных и показателей, которые обрабатываются в системе, в данном меню можно выбрать раздел для дальнейшей работы. Пользователю необходимо нажать на соответствующую картинку и перейти в другой раздел (см.рисунок 8.2)

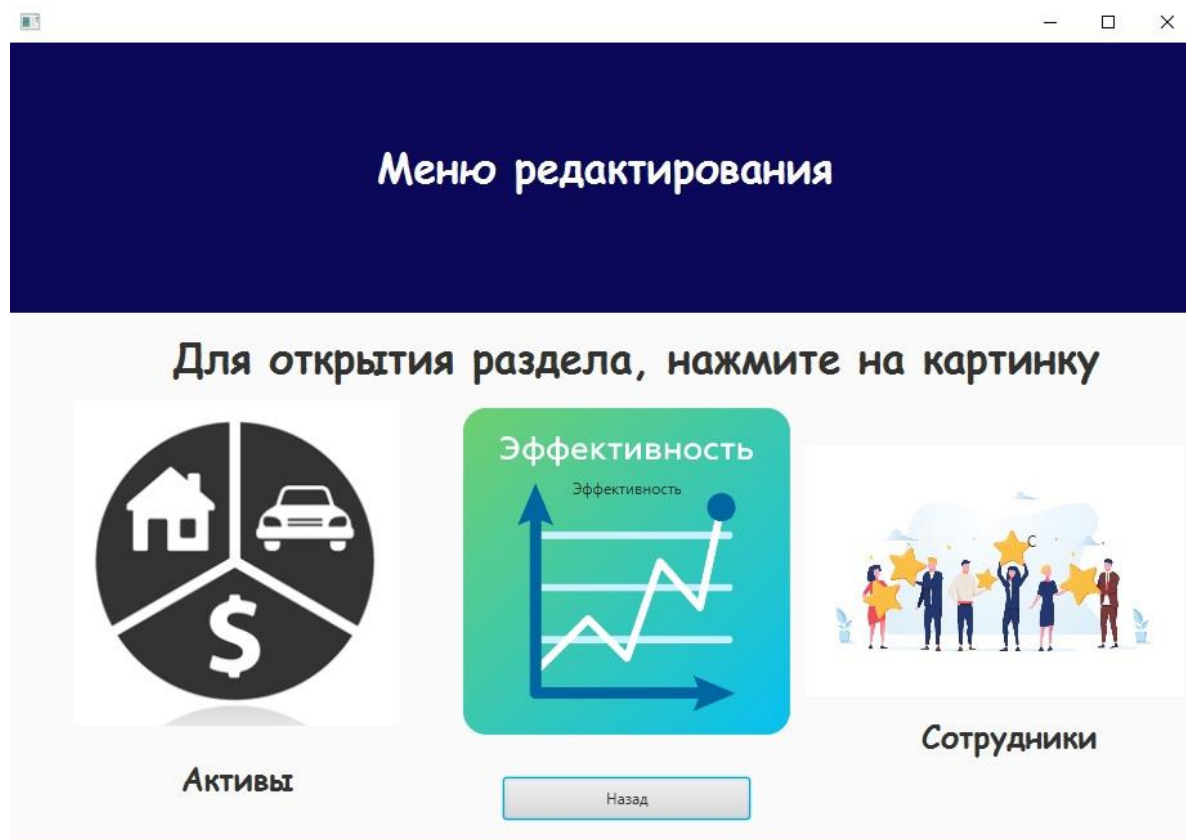


Рисунок 8.2 – Меню редактирования данных

Помимо этого, данные в системе можно сохранять в отчет (см.рисунок 8.3)

Файл	Правка	Формат	Вид	Справка
ID	Актив	Окупаемость	Рентабельность	Суммарная выгода
43	Принтер №1	0.7	143.0	27900.0
44	Компьютер №3	0.43	233.0	145685.0
45	Компьютер №4	0.303	330.0	77391.0
46	Лицензия JK	1800.0	0.0	-1799.0
47	GeForce RTX 3090	5.333	19.0	7001.0
48	AMD Radeon RX 6900 XT	9.909	10.0	4087.0

Рисунок 8.3 – Меню редактирования данных

Отчет необходим для дальнейшей работы с данными, и сохраняется в виде текстового документа с основными данные.

Во время работы приложения могут возникать определенные ошибки, связанные с различными факторами, как человеческими, так и системными. Для корректной работы системы, необходимо реализовать определенную обработку исключительных ситуаций.

Если возникает необходимость корректировать данные, которые уже находятся в системе, администратор переходит в соответствующее меню и переходит в раздел данных для изменения нужных данных (см.рисунок 8.4)

Рисунок 8.4 – Окно редактирования показателей эффективности

Пользователю необходимо выбрать, какие данные нужно изменить, после чего выбрать интересующий актив, путем нажатия на верхний пункт. После выбора актива, все поля, связанные с ним, будут перенесены в интерфейс (см.рисунок 8.5).

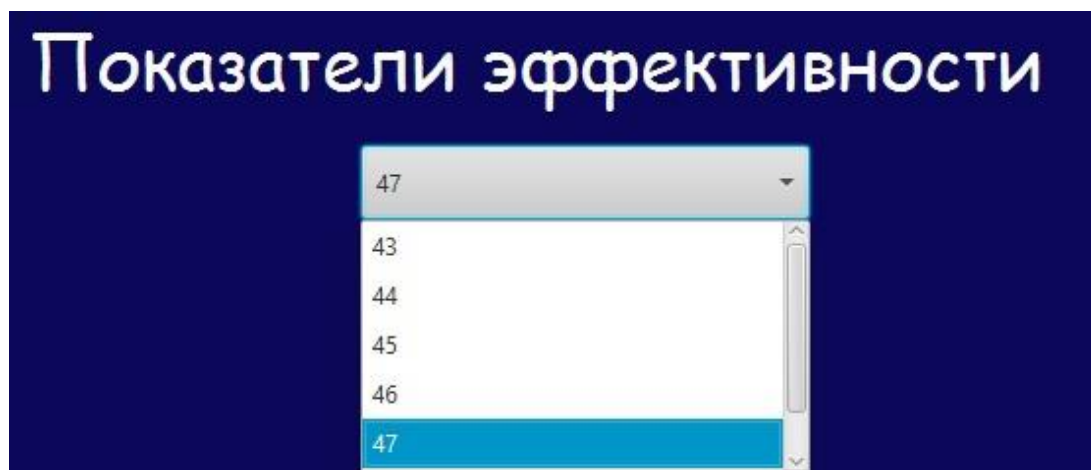


Рисунок 8.5 – Выбор актива для редактирования

После выбора актива из поля, данные переносятся в интерфейс для дальнейшей работы (см.рисунок 8.6)

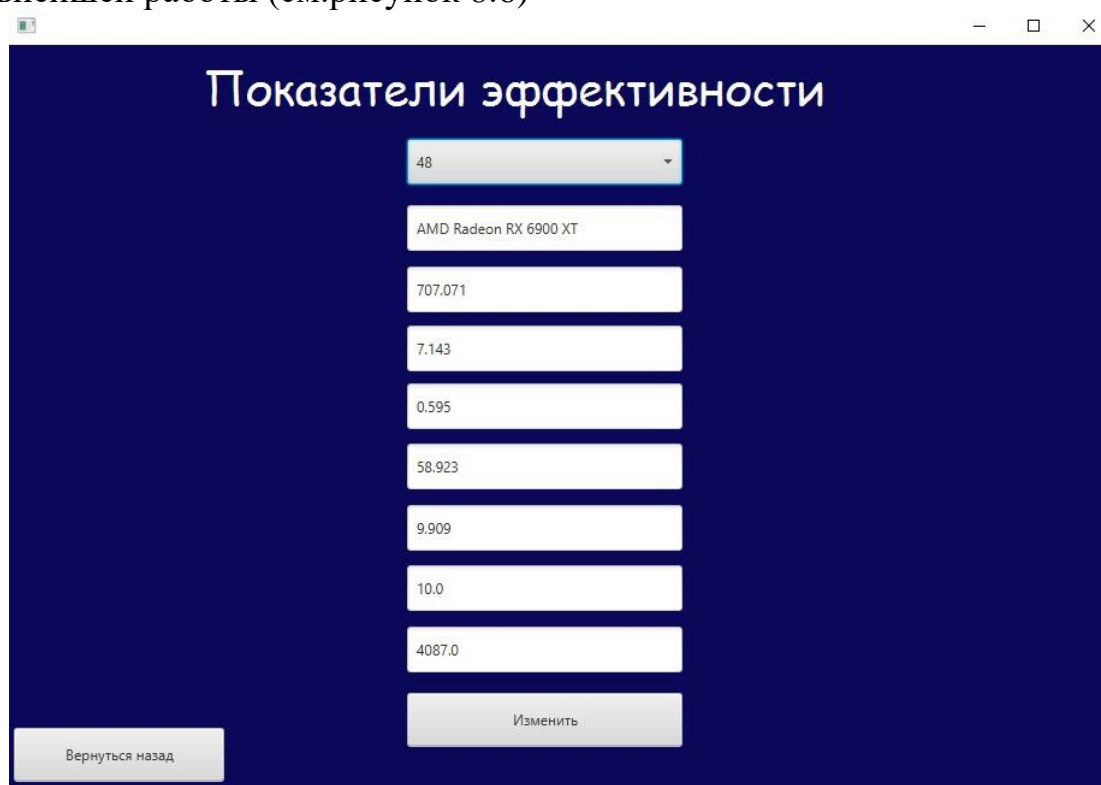


Рисунок 8.6 – Вывод данных для редактирования

В разделе просмотра данных, отображается информация по всем активам, помимо этого, данные можно сортировать по определенным параметрам, для этого необходимо нажать на кнопку заголовка данных и они автоматически распределятся либо в порядке возрастания, либо убывания (см.рисунок 8.7).



Помимо просмотров, если возникает необходимость убрать данные, это можно сделать в этом же разделе. В интерфейсе указаны подсказки, как реализовать данное действие. Для этого пользователю необходимо выбрать интересующий актив, нажав на него, после чего кликнуть на корзину. При этом если не выбрать данные, но нажать на корзину, данные останутся в системе, что предотвратит случайное действие.

ЗАКЛЮЧЕНИЕ

Итогом данного курсового проекта является функционирующее приложение, предназначенное для облегчения и удобства в работе ИТ-активами предприятия.

Приложение разработано на объектно-ориентированном языке программирования Java. Для создания графического пользовательского интерфейса была использована библиотека JavaFX.

Данные, с которыми работает приложение, записываются и хранятся в базе данных, которая связана с СУБД MySQL с помощью менеджера баз данных JDBC.

Приложение является клиент-серверным и поддерживает многопоточность, поэтому несколько пользователей могут одновременно работать в приложении.

В ходе выполнения данного курсового проекта была изучена предметная область и выявлены причины внедрения информационной системы. Для наглядности предметной области была построена информационная модель IDEF0, а также диаграммы UML. Помимо этого, в ходе выполнения проекта было составлено руководство пользователю, где простым и доступным языком описывается принцип работы программы. В завершение работы было проведено тестирование разработанной системы, подтвердившее работоспособность созданного программного средства.

Основные функции программного средства реализованы в соответствии с выявленными особенностями предметной области. Был разработан довольно широкий функционал для работы с информацией, которая содержится в базе данных. Интерфейс программы был разработан для легкого и доступного использования приложения любым пользователем без особых временных затрат.

Бизнес-логика системы даёт возможность анализировать данные активов, а также рассчитывать их эффективность, визуализировать основные показатели на диаграммах для более комфортной работы с данными.

Таким образом, можно сделать вывод, что поставленные в начале работы цели и задачи были выполнены, программа соответствует необходимым нормам и готова к своему использованию на практике. В дальнейшем приложение может быть усовершенствовано в соответствии с новыми идеями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] MySQL [Электронный ресурс]. – Режим доступа: <https://metanit.com/sql/mysql>.
- [2] Руководство по языку программирования Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/>
- [3] Основы управления ИТ-активами [Электронный ресурс]. – Режим доступа: <https://ibs.ru/products/IT-infrastructure-service-management>.
- [4] Активы ИТ-компаний [Электронный ресурс]. – Режим доступа: <https://it-guild.com/info/blog/upravlyat-aktivami-itam-ili-vesti-ambarnuyu-knigu>.
- [5] Эффективное управление ИТ-активами [Электронный ресурс]. – Режим доступа: <https://freshservice.com/ru/features/it-asset-management>.
- [6] Unified Modeling Language [Электронный ресурс]. – Режим доступа: <https://www.visual-paradigm.com/guide/>.
- [7] Введение в MVC и HMVC [Электронный ресурс]. – Режим доступа: <https://ruseller.com/>
- [8] Основы баз данных [Электронный ресурс]. – Режим доступа: <https://www.site-do.ru/db/db1.php>
- [9] YouTube [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.youtube.com/user/syssoftga>.
- [10] Проектирование базы данных [Электронный ресурс]. – Режим доступа: http://www.bseu.by/it/tohod/lekcii4_3.htm. – Дата доступа: 15.10.2021
- [11] Введение в MVC и HMVC [Электронный ресурс]. – Режим доступа: <https://ruseller.com/>
- [12] Черемных, С. Моделирование и анализ систем. IDEF-технологии Черемных С.В., Семенов И.О., Ручкин В.С, Москва: Россия. – 2006. – 192 с.
- [13] Основы ИТ-активов [Электронный ресурс]. – Режим доступа: <https://cleverics.ru/ideas/asset-management-essentials>.
- [14] Управление ИТ-активами [Электронный ресурс]. – Режим доступа: <https://www.ibm.com/ru-ru/topics/enterprise-asset-management>.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг алгоритмов, реализующих бизнес-логику

```
public class ClientHandler {
private ServerTCP server;
private Socket socket;
private ObjectInputStream in;
private ObjectOutputStream out;
private DatabaseConnection db;
static int counter = 0;

public ClientHandler(ServerTCP srv, Socket sock) throws
ClassNotFoundException {
try {
this.server = srv;
this.socket = sock;
this.db = new DatabaseConnection();
this.out = new ObjectOutputStream(socket.getOutputStream());
this.in = new ObjectInputStream(socket.getInputStream());
new Thread() -> {
counter++;
System.out.println("Подключение №" + counter + " Адрес: " +
socket.getInetAddress() + ", Порт:" + socket.getPort() + "(" +
socket.getLocalPort() + ")");
try {
DataObject dataObj;
try {
dataObj = (DataObject) in.readObject();
switch (dataObj.getCommand()) {
case "adminAuthorization": {
dataObj = this.db.getAuthorization(dataObj, DatabaseOptions.ADMIN_TABLE);
sendMsg(dataObj);
break;
}
case "userAuthorization": {
dataObj = this.db.getAuthorization(dataObj, DatabaseOptions.USER_TABLE);
sendMsg(dataObj);
break;
}
case "adminRegistration": {
dataObj = this.db.getAuthorization(dataObj, DatabaseOptions.ADMIN_TABLE);
if (!dataObj.getResult()) {
```



```

dataObj = this.db.getRegistration(dataObj, DatabaseOptions.ADMIN_TABLE);
}
sendMsg(dataObj);
break;
}
case "userRegistration": {
dataObj = this.db.getAuthorization(dataObj, DatabaseOptions.USER_TABLE);
if (!dataObj.getResult()) {
dataObj = this.db.getRegistration(dataObj, DatabaseOptions.USER_TABLE);
}
sendMsg(dataObj);
break;
}
case "deleteUser": {
this.db.deleteUser(dataObj, DatabaseOptions.USER_TABLE);
sendMsg(dataObj);
break;
}
case "showAssets": {
ArrayList<DataObject> objectArrayList = null;
objectArrayList = this.db.getAssets(objectArrayList);
sendObj(objectArrayList);
break;
}
case "editAssets": {
this.db.editAssets(dataObj);
sendMsg(dataObj);
break;
}
case "addAssets": {
dataObj = this.db.getAsset(dataObj);
sendMsg(dataObj);
break;
}
case "deleteAssets": {
this.db.deleteAssets(dataObj);
sendMsg(dataObj);
break;
}
case "addDepreciation": {
dataObj = this.db.getDepreciation(dataObj);
sendMsg(dataObj);
break;
}
}

```

```

case "showDepreciation": {
    ArrayList<DataObject> objectArrayList = null;
    objectArrayList = this.db.getDepreciation(objectArrayList);
    sendObj(objectArrayList);
    break;
}
case "deleteDepreciation": {
    this.db.deleteDepreciation(dataObj);
    sendMsg(dataObj);
    break;
}
case "editDepreciation": {
    this.db.editDepreciation(dataObj);
    sendMsg(dataObj);
    break;
}
// Сотрудники
case "addEmployeeData": {
    dataObj = this.db.getEmployee(dataObj);
    sendMsg(dataObj);
    break;
}
case "showEmployee": {
    ArrayList<DataObject> objectArrayList = null;
    objectArrayList = this.db.getEmployee(objectArrayList);
    sendObj(objectArrayList);
    break;
}

case "deleteEmployee": {
    this.db.deleteEmployee(dataObj);
    sendMsg(dataObj);
    break;
}

case "editEmployee": {
    this.db.editEmployee(dataObj);
    sendMsg(dataObj);
    break;
}
}
System.out.println("Операция прошла успешно");
} catch (EOFException e) {

```

```

} catch (SocketException e) {

    public class AdminMenuController {
private ObservableList<DataObject> depreciation =
FXCollections.observableArrayList();
private DataObject dataObject;
private ArrayList<DataObject> dataObjects;
@FXML
private ResourceBundle resources;
@FXML
private URL location;
@FXML
private Button showBtn;
@FXML
void toAddMenu(ActionEvent event) throws IOException { // функция
запускающаяся при нажатии на кнопку
Client.setRoot("AdminMenuAdd"); // Навзание fxml куда будет переход
}
@FXML
void toEditMenu(ActionEvent event) throws IOException {
Client.setRoot("AdminMenuEdit");
}
}
@FXML
void toGraphic(ActionEvent event) throws IOException {
Client.setRoot("AdminMenuGraphicShow");
}
@FXML
void onBack(ActionEvent event) throws IOException {
Client.setRoot("Authorization");
}

}
@FXML
void toAddMenuEmployee(ActionEvent event) throws IOException {
Client.setRoot("AdminMenuAddEmployee");
}

@FXML
void fileRes(ActionEvent event) throws IOException, ClassNotFoundException {
DatabaseConnection fr = new DatabaseConnection();
fr.filewriter();
}

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг основных элементов программы

```
public class Connection {
    protected static String current;
    protected static int current_id;

    private static final String IP = "127.0.0.1";
    private static final int PORT = 1010;
    protected Socket socket;
    protected ObjectOutputStream out;
    protected ObjectInputStream in;
    protected void connect() {
        try {
            socket = new Socket(IP,PORT);
            in = new ObjectInputStream(socket.getInputStream());
            out = new ObjectOutputStream(socket.getOutputStream());
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    protected void closeConnect() {
        try {
            socket.close();
            in.close();
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public class ServerTCP {
    protected static int PORT=1010;
    public ServerTCP() {
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {
            System.out.println("Сервер успешно запущен!");
            while (true) {
                Socket socket = serverSocket.accept();
                new ClientHandler(this,socket);
            }
        }
    }
}
```

```

    }
    } catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
    }
    }
    public class AdminMenuSDController {
    @FXML
    private ResourceBundle resources;
    @FXML
    private URL location;
    @FXML
    void onBack(ActionEvent event) throws IOException {
    Client.setRoot("AdminMenu");
    }
    @FXML
    void toDeleteAssets(ActionEvent event) throws IOException {
    Client.setRoot("AdminMenuSDAssets");
    }
    @FXML
    void toDeleteDepreciation(ActionEvent event) throws IOException {
    Client.setRoot("AdminMenuSDDepreciation");
    }
    @FXML
    void toDeleteEmployee(ActionEvent event) throws IOException {
    Client.setRoot("AdminMenuSDEmployee");
    }
    @FXML
    void initialize() {
    }
    }

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

```
create database if not exists `amort`;
use `amort`;
create table if not exists `admins` (
`id` int AUTO_INCREMENT PRIMARY KEY,
`login` varchar(40) not null,
`password` varchar(40) not null
) DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
create table if not exists `users` (
`id` int AUTO_INCREMENT PRIMARY KEY,
`login` varchar(40) not null,
`password` varchar(40) not null
) DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
create table if not exists `assets` (
`id` int AUTO_INCREMENT PRIMARY KEY,
`name` varchar(30),
`price` int not null,
`termOfUse` int not null
) DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
create table if not exists `depreciation` (
`id` int AUTO_INCREMENT PRIMARY KEY,
`name` varchar(30),
`yearPercent` double not null,
`yearPrice` double not null,
`monthPercent` double not null,
`monthPrice` double not null,
`payBack` double not null,
`rentability` double not null,
`incomeMoney` double not null
) DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
create table if not exists `employees` (
`id` int AUTO_INCREMENT PRIMARY KEY,
`name` varchar(30),
`surname` varchar(30),
`department` varchar(30)
`idAssets` int not null
) DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```