Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Институт информационных технологий

Специальность: Информационные системы и технологии (в обеспечении промышленной безопасности)

**Пояснительная записка**

**КУРСОВАЯ РАБОТА**

по курсу «Программирование сетевых приложений»

Тема: «Инструменты управления проектами в компании и программная поддержка количественной оценки потенциальных рисков IT-проекта»

Студент-вечерник 2 курса

группы № 980161

ФИО: Алейчик И.Д.

Адрес: г. Минск ул. Каролинская д.8

тел. +375(33) 334-31-56

Минск, 2020

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc473633993)

[1 Описание процесса амортизации основных средств 5](#_Toc473633994)

[1.1 Общие понятия и термины 5](#_Toc473633995)

[1.2 Определение процессов, требующих автоматизации 8](#_Toc473633996)

[2 Постановка задачи и обзор методов ее решения 10](#_Toc473633997)

[3 Функциональная  модель    учёта амортизации основных средств предприятия 13](#_Toc473633998)

[4 Построение информационной модели 17](#_Toc473633999)

[5 Спецификация вариантов использования системы 19](#_Toc473634000)

[6 Модели представления системы и их описание 21](#_Toc473634001)

[6.1 Диаграммы состояний (Statechart diagram) 21](#_Toc473634002)

[6.2 Диаграмма последовательности регистрации нового пользователя (Sequence diagram) 21](#_Toc473634003)

[6.3 Диаграммы классов 22](#_Toc473634004)

[6.4 Диаграммы компонентов (component diagram) 22](#_Toc473634005)

[6.5 Диаграмма развертывания (deployment diagram) 23](#_Toc473634006)

[7 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 24](#_Toc473634007)

[8 Руководство пользователя по развертыванию системы 25](#_Toc473634008)

[9 Результаты тестирования системы учёта амортизации основных средств 27](#_Toc473634009)

[Заключение 39](#_Toc473634010)

[Список использованных источников 40](#_Toc473634011)

[Приложение А(обязательное)Модели представления системы(к разделу 6) 41](#_Toc473634012)

[Приложение Б (обязательное) Блок-схемы программы (к разделу 7) 46](#_Toc473634015)

[Приложение В (обязательное) Скрипт создания БД 48](#_Toc473634018)

[Приложение Г (обязательное) Выборочный листинг кода 50](#_Toc473634021)

# Введение

В настоящее время во всех отраслях человеческой деятельности огромное внимание уделяется использованию информационных технологий для наиболее рационального и эффективного использования ресурсов, имеющихся в распоряжении человека.

В современных условиях эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Следовательно, повышение эффективности становится одним из направлений совершенствования деятельности в целом. Наиболее очевидным способом повышения эффективности протекания трудового процесса является его автоматизация. Бурное развитие информационных компьютерных технологий, совершенствование технической платформы и появление принципиально новых классов программных продуктов привело к изменению подходов к автоматизации управления предприятиями, автоматизации документооборота, автоматизации различных процессов предприятий.

Начисление амортизации основных средств обычно не представляет никаких методических трудностей, однако на практике обычно представляет собой однообразную, повторяющуюся из месяца в месяц операцию. В отдельных случаях бухгалтеру могут потребоваться данные по сумме износа, начисленного по отдельному объекту основных средств, его первоначальной (восстановительной) и остаточной стоимости. Кроме того необходимо контролировать сумму начисленного износа, чтобы не допустить его излишнее начисление. Так, основным результатом автоматизации этого участка учета должно стать освобождение бухгалтера от однообразной работы. Программа бухгалтерского учета должна также позволять формировать различные отчеты и вести аналитический учет в разрезе отдельных объектов основных средств.

ПЭВМ при рассмотрении данных вопросов позволяет принимать более эффективные управленческие решения, снижает трудоемкость работников, обеспечивает более удобный и быстрый доступ к необходимой информации. Однако решение этих задач невозможно без создания автоматизированных информационных систем и программных средств, позволяющих накапливать, обрабатывать и выдавать требуемую информацию в диалоговом режиме.

Каждое производственная организация имеет в своем распоряжении основные и оборотные средства. Поэтому основной целью является создание базы данных, которая позволит структурировать те огромные объемы

информации, которые накапливаются на предприятиях, и тем самым существенно облегчить работу и доступ к данным практически всем звеньям, участвующим в управлении основными средствами предприятия.

Таким образом, целью данного курсового проекта является сокращение временных затрат сотрудников бухгалтерских служб за счёт автоматизации обработки необходимой информации: различных расчётов, процесса составления отчетности, представления расчётов 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;
* реализовать серверную часть, имеющую доступ к базе данных и выполняющую ряд необходимых действий, позволяющих достигнуть цель курсового проекта;
* реализовать клиентскую часть с дружественным интерфейсом, обеспечивающий максимально удобное взаимодействие с программой или вычислительной системой.

# 1 Описание предметной облости

## 1.1 Общие понятия и термины

**Перед тем, как рассмотреть понятие амортизация, разберём сущность, виды и классификацию основных средств предприятия.**

**Основные средства** – это товарно-материальные ценности, которые в течение длительного периода времени сохраняют свою вещественно-натуральную форму, изнашиваются постепенно и переносят свою стоимость на готовую продукцию или услуги по частям.

К основным средствам предприятия (материальным основным фондам) относятся следующие их виды:

* здания (кроме жилых);
* сооружения;
* машины и оборудование;
* транспортные средства;
* производственный и хозяйственный инвентарь и принадлежности;
* рабочий, продуктивный и племенной скот;
* многолетние насаждения и прочие основные фонды.

Для первоначальной оценки основных средств используют стоимость их приобретения или создания. Поскольку основные фонды участвуют в процессе производства длительное время и постепенно изнашиваются, а условия воспроизводства за этот период меняются, существует несколько видов их стоимости:

* первоначальная стоимость;
* остаточная стоимость;
* восстановительная стоимость;
* ликвидационная стоимость.

Первоначальная стоимость – фактическая стоимость приобретения оборудования включая затраты на доставку и монтаж. Она не позволяет достоверно анализировать динамику основных фондов, поскольку, приобретение разных объектов произведенных в разное время, обуславливает то, что одинаковые по своим свойствам объекты имеют различную первоначальную стоимость.

Остаточная стоимость – соответствует первоначальной стоимости за вычетом суммы износа, образовавшемуся к этому периоду времени. Выражает стоимость основных фондов, еще не перенесенную на изготовленную продукцию.

Восстановительная стоимость – это стоимость воспроизводства основных фондов в новых производственных условиях (данного года), она определяется в процессе переоценки.

Ликвидационная стоимость – это та цена, которую предприятие рассчитывает получить вследствие реализации данной части капитала по истечении срока его службы. Под сроком службы понимается период, на протяжении которого предприятие планирует использовать данную часть капитала[1].

**Срок полезного использования** – это период, в течение которого использование объектов основных средств предприятия должно приносить доход организации или служить для выполнения целей ее деятельности. В ходе эксплуатации основные средства предприятия подвергаются износу.

Существует моральный и физический износ.

**Моральный износ** – потеря зданиями, сооружениями, машинами, автоматами и другим оборудованием своей стоимости вследствие научно-технического прогресса и роста производительности труда.

**Физический износ** происходит в результате активной работы оборудования, а также под влиянием естественных сил природы.

Единицей бухгалтерского учета основных средств предприятия является инвентарный объект со всеми приспособлениями и принадлежностями или отдельный конструктивно обособленный предмет. Основные средства предприятия принимаются к бухгалтерскому учету по первоначальной стоимости, т.е. по сумме фактических затрат на приобретение, сооружение и изготовление объекта основных средств[2].

Амортизация – это процесс постепенного переноса стоимости основных фондов на вновь созданную продукцию для последующего их восстановления (полной замены). **Другими словами, амортизация – это процесс самовозобновления фондов**: она фиксирует степень их износа и одновременно накапливает денежные средства для полного обновления.

Процесс амортизации осуществляется с помощью ежегодных амортизационных отчислений.

Амортизационные отчисления – это денежные средства, входящие в стоимость готовой продукции и предназначенные для полного восстановления основных производственных фондов. Амортизационные отчисления постепенно накапливаются и составляют амортизационный фонд, который используется для приобретения новых основных средств.

Сумма годовых амортизационных отчислений рассчитывается по каждому виду основного средства одним из следующих способов:

* линейным способом;
* нелинейным способом;
* производительным способом.

При начислении амортизации можно использовать любой из этих методов. Для этого все основные средства делят на однородные группы, у которых есть общие признаки. Например, здания, компьютеры, транспорт, мебель и т.д.

По основным средствам одной группы можно использовать только один из перечисленных методов.

Выбранный метод применяется в течение всего срока службы (полезного использования) основного средства[4].

Линейный способ амортизации предполагает равномерный перенос стоимости фондов на стоимость готовой продукции в течение всего срока их полезного использования.

Величина ежегодных амортизационных отчислений при линейном способе определяется на основании годовых норм амортизации, установленных в процентах к первоначальной стоимости **основных производственных средств**. Нормы устанавливаются предприятием самостоятельно и дифференцированы по видам основных фондов.

**Годовая норма амортизации по определенному виду основных производственных средств** рассчитывается по формуле:

где – годовая норма амортизации, %;

Т – срок полезного использования основного средства, лет.

Нелинейный способ амортизации заключается в неравномерном начислении амортизации в течение срока полезного использования основных производственных фондов.

При нелинейном способе используются следующие методы:

* метод суммы чисел лет;
* метод уменьшаемого остатка.

При использовании метода суммы чисел лет годовая сумма амортизационных отчислений определяется по каждому объекту по формуле:

где Ф – амортизируемая стоимость основного средства, руб.;

– число лет, оставшихся до конца срока полезного использования объекта, лет;

СЧЛ – сумма чисел лет срока полезного использования объекта.

Сумма чисел лет срока полезного использования объекта:

где – срок полезного использования объекта, лет.

При использовании метода уменьшаемого остатка годовая сумма амортизационных отчислений определяется произведением недоамортизированной стоимости объекта по состоянию на начало года и нормы амортизации, исчисленной исходя из срока полезного использования объекта и коэффициента ускорения (до 2,5 раз), принятого на предприятии.

Недоамортизированная стоимость объекта определяется разностью амортизируемой (полной) стоимости объекта и суммы амортизации, начисленной до начала планируемого года.

Производительный способ амортизационных отчислений заключается в начислении амортизации исходя из амортизируемой стоимости объекта и отношения натуральных показателей объема товаров (работ, услуг), выпущенных в году t, ко всему ресурсу объекта.

Амортизационные отчисления в таком случае рассчитываются по формуле:

где – сумма амортизацтонных отчислений в году t, руб.;

– прогнозируемый в течение срока эксплуатации объекта объем продукции в году t;

t = 1,…,n – годы срока полезного использования объекта[3].

## 1.2 Определение процессов, требующих автоматизации

Главной целью любой системы автоматизации является сокращение издержек и увеличение производительности работы организации. Также использование такой системы должно улучшить контроль над финансовой деятельностью предприятия, что в свою очередь увеличит эффективность управления компанией, эффективность её работы.

Важной задачей является верный расчёт амортизации для каждого основного производственного фонда. Даже самый грамотный специалист под воздействием человеческого фактора может допустить ошибку, которая в скором приведет к несоответствиям в финансах предприятия. Автоматизация учёта амортизации позволит сократить риск таких ошибок.

В результате анализа процесса вычисления амортизационных исчислений был сделан вывод о необходимости автоматизации данного процесса, так как это в разы ускорит затраты времени бухгалтера.Также работу сотрудников смогут облегчить автоматизированные процессы составления отчётов и построения диаграмм.

Таким образом, необходимо создать программный продукт, который будет отвечать следующим требованиям:

* сотрудник должен затрачивать минимальное количество временных ресурсов на различные расчёты и составление отчётов;
* интерфейс должен быть удобным и понятным для пользователя;
* должна быть создана база данных для быстрого и удобного поиска интересующей информации.

Результат поиска будет содержать информацию по каждому основному производственному средству.

# 2 Постановка задачи и обзор методов ее решения

В данном курсовомпроекте необходимо решить проблему автоматизации учёта амортизации основных средств производства. Сумма годовых амортизационных отчислений рассчитывается по каждому виду основного средства одним из следующих способов:

* линейным способом;
* нелинейным способом;
* производительным способом.

Формулы для каждого из способов были описаны выше в пункте 1.1.

При начислении амортизации можно использовать любой из этих методов. Для этого все основные средства делят на однородные группы, у которых есть общие признаки. По основным средствам одной группы можно использовать только один из перечисленных методов.

Для вычисления необходимо знать следующие параметры:

* годовую норму амортизации;
* срок полезного использования основного средства;
* амортизируемая стоимость основного средства;
* число лет, оставшихся до конца срока полезного использования объекта.

Также при сознании курсового проекта были использованы следующие программные средства.

Java – полностью объектно-ориентированный язык программирования. Java используется для разработки программного обеспечения проектов различного назначения: начиная от приложений для мобильного телефона, заканчивая масштабными корпоративными приложениями. Java может использоваться для создания двух типов программ: приложений и апплетов. Приложения Java могут быть непосредственно выполнены, используя интерпретатор Java. Апплет - небольшая программа работающая с окнами, которые внедрены в страницу HTML.

Встроенная поддержка многопоточности снабжает программистов Java мощным инструментом для улучшения интерактивной работы графических приложений. Потоки иногда также называют легкими процессами или контекстами выполнения. Библиотека Java обеспечивает класс потока, который содержит обширную коллекцию методов запуска, выполнения, и остановки потока, а также проверки его состояния[5].

Самой выдающейся чертой языка Java является то, что его приложения компилируются в специальный байт-код и выполняются в специальной виртуальной машине в любой операционной системе и на любом оборудовании.

Другим важным преимуществом языка является довольно высокая безопасность. Поскольку все приложения выполняются и, соответственно, контролируются Java-машиной, то при любой попытке несанкционированного доступа или соединения программа просто прерывает свою работу. Некоторые программисты выделяют и более упрощённую саму форму написания программ Java по сравнению с её предшественниками, коими являются С, С+ и С++. Помимо этого к языку существует огромное количество бесплатных программных библиотек, написанных программистами всего мира[6].

Также Java обеспечивает автоматическое освобождение неиспользуемой памяти. Наряду с распределением памяти программистом JVM ведет учет всех используемых блоков памяти и указателей на них. Если блок памяти больше не используется, он может быть освобожден. Это обеспечивает процесс, который называется "сборкой мусора". Он периодически вызывается JVM, проверяет все используемые блоки памяти и освобождает те из них, на которые отсутствуют указатели. Сборка мусора очень удобна, но за ее использование приходится расплачиваться большим потреблением памяти и низкой производительностью[7].

Данный курсовой проект реализован с помощью клиент-серверной архитектуры.

Технология «клиент-сервер» дает большую безопасность для пользователя, процессы выполняются более стабильно, согласованно, с повышенной конфиденциальностью, данные быстро обрабатываются и надежно хранятся.

В архитектуру клиент-сервер входят следующие основные компоненты:

* сервер баз данных отвечает за хранение, доступ, защиту и резервное копирование данных;
* сервер приложений - это устройство, выполняющее определенные бизнес-правила;
* клиент предоставляет интерфейс пользователя;
* сеть и коммуникационное ПО – это всевозможное оборудование, каналы для передачи данных и ПО используемое для осуществления передачи запросов и ответов от клиента к серверу и обратно через сетевые протоколы.

Использование такой архитектуры помогает оптимизировать распределение вычислительных ресурсов и обеспечивает защиту данных.

Как правило, программа обработки данных (клиентская часть) располагается на одном персональном компьютере, а сама база данных - на другом. Такое разделение вполне закономерно: клиентская часть программы (та, что находится у пользователя), передает по сети запрос на обработку данных на другой компьютер, на котором сервер базы данных их читает, выполняет, и возвращает по сети ответ клиенту. Причем по сети передается лишь полезная информация.

MySQL – это система управления реляционными базами данных.

В реляционной базе данные хранятся 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. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

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).

Система безопасности MySQL основана на привилегиях и паролях с возможностью верификации с удаленного компьютера, за счет чего обеспечивается гибкость и безопасность. Пароли при передаче по сети при соединении с сервером шифруются. Клиенты могут соединяться с MySQL, используя сокеты TCP/IP, сокеты Unix или именованные каналы.

MySQL поддерживает параллельный доступ к базе данных. Приложения могут обращаться к базе данных одновременно, что повышает общую производительность системы. Кроме того, отдельные операции могут «распараллеливаться» для еще большего улучшения производительности.

MySQL помогает восстанавливать информацию в случае непредвиденного сбоя, незаметно для пользователей создавая резервные копии данных. Все изменения, вносимые в базу данных, регистрируются, поэтому многие операции можно отменять и выполнять повторно.

# 3 Функциональная модель учёта амортизации основных средств предприятия

Целью данного курсового проекта является создание программного продукта, который автоматизирует процесс расчёта амортизационных отчислений.Для этого необходимо более детально изучить полный процесс учёта амортизации основных средств бухгалтерами. Для этого была разработана модель по методологии IDEF0, контекстная диаграмма верхнего уровня которого представлена на рисунке 3.1.

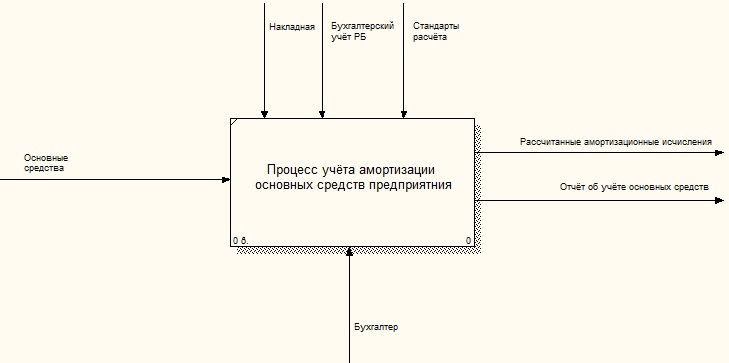


Рисунок 3.1– Контекстная диаграмма верхнего уровня

В основном процессе можно выделить следующие подпроцессы:

* определение срока полезного использования средства;
* уменьшение стоимости используемого средства;
* включение величины износа в цену товара;
* начисление величины износа на амортизационный счёт.

Декомпозиция контекстной диаграммы верхнего уровня показана на рисунке 3.2.

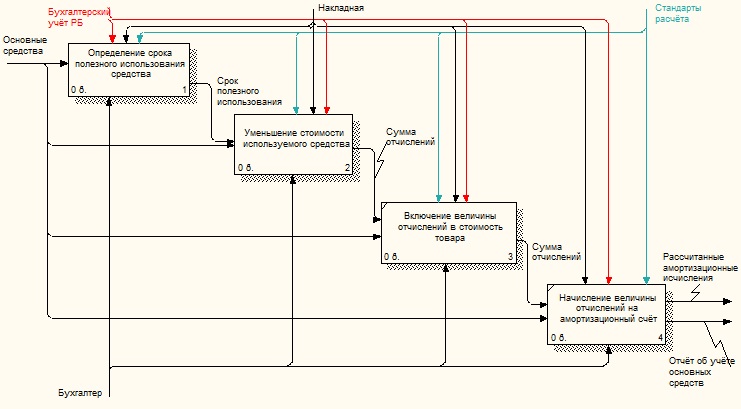


Рисунок 3.2– Декомпозиция контекстной диаграммы верхнего уровня

Процесс начинается с определения срока полезного использования основного средства. На этом этапе бухгалтер заполняет всю необходимую для дальнейших расчётов информацию. Далее, чтобы определить сумму годового износа, необходимо провести некоторые вычисления. После получения суммы отчислений, нужно включить эту величину в стоимость товара и далее зачислить на амортизационный счёт. Декомпозиция блока «Определение срока полезного использования средства» представлена на рисунке 3.3.

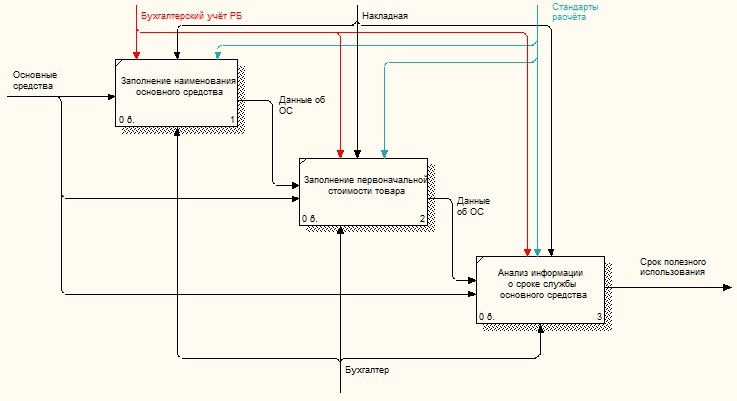


Рисунок 3.3 – Декомпозиция блока «Определение срока полезного использования средства»

На этапе «Уменьшение стоимости используемого средства» вычисляется сумма амортизационных отчислений, которая в дальнейшем будет использоваться для подсчёта стоимости товара. За счёт увеличения стоимости при реализации товара предприятие получит те самые отчисления на покупку нового оборудования через некоторое время, так же заранее определённое. Декомпозиция вышеописанного блока представлена на рисунке 3.4.

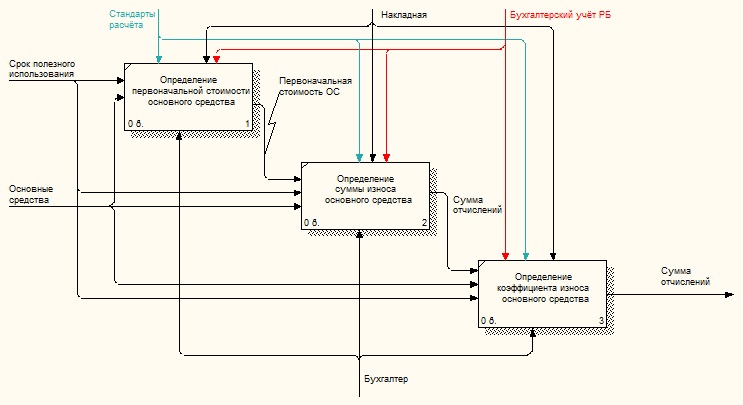


Рисунок 3.4 – Декомпозиция блока «Уменьшение стоимости используемого средства»

Рассмотрим подробнее блок «Определение суммы износа основного средства». Здесь определяются общие затраты на приобретение и обслуживание оборудование, что также учитывается при расчёте амортизационных отчислений. Декомпозиция этого блока представлена на рисунке 3.5.

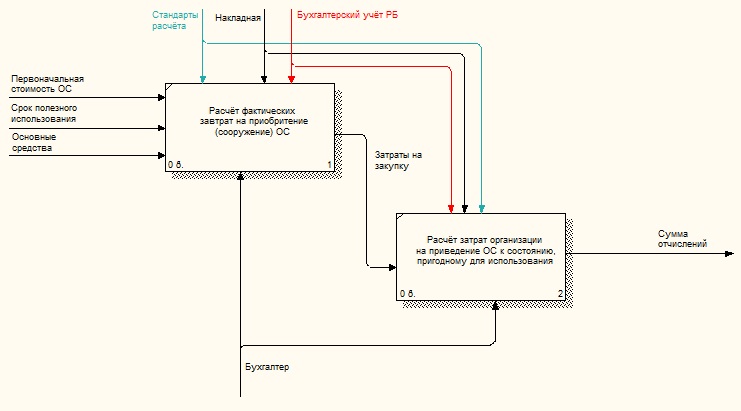


Рисунок 3.5 – Декомпозиция блока «Определение суммы износа основного средства»

# 4 Построение информационной модели

В процессе информационного моделирования были выделены следующие сущности:

* Пользователь;
* Статус;
* Основной фонд;
* Амортизация.

Сущности «Пользователь» и «Статус» необходимы для реализации процесса авторизации. Сущность «Пользователь» содержит логин и пароль каждого пользователя, а сущность «Статус» определяет, какие права доступа к данным будут предоставлены пользователю в зависимости от значения статуса. Эти сущности связаны между собой связью один-ко-многим, так как сразу несколько пользователей могут обладать одной ролью.

Сущность «Основной фонд» необходима для хранения информации об основных средствах предприятия, для которых необходимо рассчитать амортизационные отчисления. Она хранит такие данные, как название основного фонда, первоначальную стоимость основного фонда и срок полезного использования.

Сущность «Амортизация» необходима для хранения данных после выполнения расчётов по формулам из пункта 1.1. Она хранит данные о месячном и годовом проценте амортизации, а так же о месячной и годовой стоимости амортизации. Эта сущность связана с сущностью «Основной фонд» связью один-к-одному, так как для каждого основного фонда может существовать только один расчёт амортизацтонных отчислений.

С учетом выявленных связей между сущностями смоделируем их взаимодействие в формате IDEF1. Информационная модель соответствует условиям третьей нормальной формы – неключевые атрибуты зависят от ключей, причем от ключей целиком и ни от чего другого, кроме как от ключей.

Общая схема связанных сущностей базы данных представлена на рисунке 4.1.

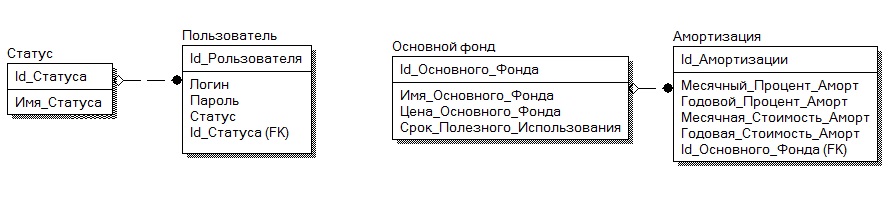


Рисунок 4.1– Информационная модель базы данных

# 5 Спецификация вариантов использования системы

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующих в процессе.

В данном курсовом проекте были определены четыре основных действующих лица: администратор, бухгалтер, пользователь и незарегистрированный пользователь.

Последнему предоставлена единственная функция регистрации в системе. Пользователь же уже знает, что он зарегистрирован и он может получить доступ к системе с правами либо администратора, либо бухгалтера.

Администратору и бухгалтеру представлены некоторые одинаковые функции, такие как:

* просмотр подробной информации об основных средствах предприятия;
* поиск необходимой информации;
* сортировка по заданным полям;
* фильтрация по заданному диапазону значений;
* создание подробного отчёта;
* построение наглядной диаграммы.

Администратору предоставлен более широкий функционал, для него реализованы возможности:

* добавление новых пользователей (сотрудиков или администраторов);
* добавление новых записей об основных средствах предприятия;
* изменение статуса пользователей;
* редактирование имеющейся информации;
* удаление записей.

При формировании отчета по данным об амортизации каждого их основных фондов создается текстовый отчет с необходимой информацией.

Диаграмма вариантов использования представлена на рисунке 5.1.

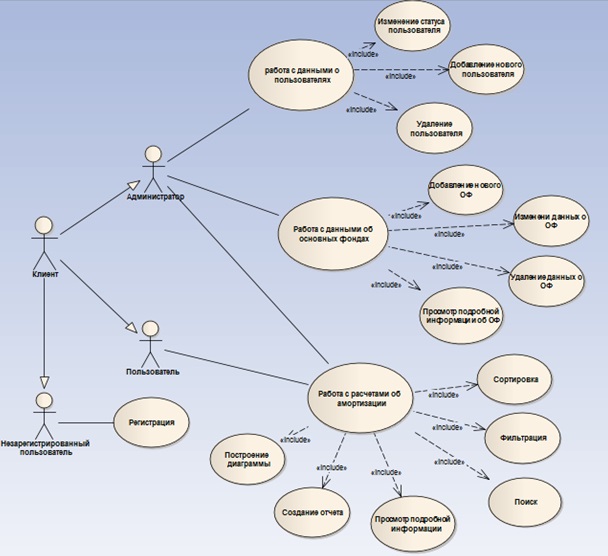


Рисунок 5.1 – Диаграмма вариантов использования (Use Case)

# 6 Модели представления системы и их описание

## 6.1 Диаграммы состояний (Statechart diagram)

Диаграмма состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Рассмотрим диаграмму на примере работы данной программы (рисунок А.1).

В начале происходит запуск приложения. Далее происходит авторизация с возможностью зарегистрироваться как бухгалтер. В случае, если авторизация прошла успешно, пользователю предоставляется доступ к ресурсам, определяемый его статусом. У администратора, более широкий функционал. У него есть возможность работать с новыми основными средствами, добавлять их в базу данных, а так же работать с пользователями.

В любом из состояний, представленных на диаграмме, каждый пользователь имеет возможность завершить сеанс работы.

## 6.2 Диаграмма последовательности регистрации нового пользователя (Sequence diagram)

Диаграмма последовательностейотносится к диаграммам взаимодействия UML, которые описывают поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами,диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами. Диаграмма приведена на рисунке А.2.

Для того, чтобы добавить нового пользователя, необходимо сначала выбрать соответствующий пункт меню. Далее необходимо ввести данные аутентификации. После этого данные отправляютсяна сервер, а сервер, в свою очередь, запрашивает данные о уже существующих пользователях в базе данных, сравнивает логин введённый новым пользователем с уже зарегистрированными. Если логины совпадают, пользователь не будет добавлен. Если же совпадений не было, то новый пользователь будет успешно добавлен и получит подтверждение от сервера.

## 6.3 Диаграммы классов

Так как это приложение создано с помощью клиент-серверной архитектуры, для клиента и для сервера будут определены свои отдельные диаграммы классов. Изображенные на рисунках A.3.1 – А.3.3.

Рассмотрим классы клиентского приложения.

Пакет Client содержит два класса – Client и SocketStream. В них реализовано соединение с сервером, получение сообщений с клавиатуры, обработка информации и отправление этих данных на сервер.

Пакет Data содержит три класса – Asset, Depreciation и User. Класс Asset содержит данные об основных фондах, а также конструкторы, геттеры и сеттеры. Класс User содержит логины и пароли пользоватей, их статус, а также конструкторы и методы get() и set(). Класс Depreciation содержит данные по вычислению амортизации, а также геттеры и сеттеры.

Пакет Forms содержит элементы графического интерфейса и методы-обработчики событий по нажатию кнопок.

Пакет TableModels включает в себя три класса – AssetModel, DepreciationModel и UserModel. Они реализованы для корректного и аккуратного представления всех данных из классов Asset, Depreciation и User в табличном виде.

Рассмотрим классы серверного приложения.

Пакет Server содержит три класса – DoIt, Server и ServerThread. Два последних класса содержат методы, позволяющие клиенту подключаться и передавать данные. В классе DoIt содержатся методы, позволяющие эти данные обрабатывать. Также в серверной части приложения есть пакет DB, который содержит класс MyDB. Этот класс позволяет подключаться к базе данных, делать выборки, добавлять, удалять и обновлять в ней информацию.

## 6.4 Диаграммы компонентов (component diagram)

Диаграммы компонентов показывают, как выглядит модель системы на физическом уровне. На диаграмме изображены компоненты программного обеспечения и связи между ними. При этом выделяют два типа компонентов: исполняемые компоненты и библиотеки кода.

На сервере находятся пакеты DBи Server c находящимися внутри классами. Необходимым компонентом является mysql-connector-java-5.1.39-bin.jar, который позволил создать соединение с базой данных MySQL.

На клиентской части содержится несколько пакетов: Client, Data, User, каждый из которых содержит несколько компонентов-классов.

## 6.5 Диаграмма развертывания (deployment diagram)

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. Диаграмма развертывания является единой для системы в целом, поскольку должна всецело отражать особенности ее реализации.

Данная диаграмма для разработанной системы представлена на рисунке А.5. Она состоит из ПК клиента и ПК сервера, которые связаны с помощью протокола TCP/IP. Также она включает базу данных, которая хранит все данные. Только сервер имеет доступ к базе данных.

# 7 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы

Рассмотрим алгоритм расчёта амортизационных отчислений, так как это является основной частью данного курсового проекта. Для того, чтобы рассчитать месячные и годовые проценты, месячную и годовую стоимость, необходимы уже некоторые данные по основным фондам в базе данных.

В клиентской части реализована функция, которая запрашивает данные об основных фондах у сервера. Сервер, имея доступ к базе данных, получает эти данные с помощью класса ResultSet и передаёт их клиенту. Клиент, в свою очередь, получив данные, подставляет их в формулу линейного определения амортизации. После чего отправляет рассчитанные данные серверу, а сервер добавляет новые записи в базу данных. Это основная бизнес-функция приложения.

Рассмотрим ещё один алгоритм: добавление нового пользователя которую может выполнить администратор.

Администратор нажимает кнопку добавления, после чего появляется панель регистрации, где необходимо ввести логин, а затем пароль два раза. Если все данные корректно введены (пароль должен содержать не менее шести символов), данные отправляются на сервер. Далее сервер запрашивает в базе данных выборку логинов уже зарегистрированных пользователей и сравнивает их с полученным. В случае, если совпадение произошло, сервер возвращает клиенту сообщение об ошибке. В противном случае, сервер продолжает проверку – сравнивает два последовательно введённых пароля на совпадение. Если пароли совпали – сервер отправляет клиенту сообщение об успешной регистрации, и делает запрос в базу данных о добавлении нового пользователя. Если же пароли не совпадают, клиент получит сообщение о несовпадении, которое отобразится на экране.

# 8 Руководство пользователя по развертыванию системы

Чтобы развернуть систему , необходимо установить PostgreSQL 13.0. После установки открыть базу данных и запустить готовый скрипт. Так же необходимо наличие JavaVirtualMachine. Для запуска программы нужно выполнить файл ProjectManager.jar..

После запуска этого файла будет отображен интерфейс, на котором нужно будет вывести все необходимые данные.

Скрипт генерации базы данных:

CREATETABLE `user` (

`iduser` INT(11) NOT NULL AUTO\_INCREMENT,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`status` INT(2) NOT NULL DEFAULT '0',

PRIMARY KEY (`iduser`)

);

CREATE TABLE `employes` (

`ideml` INT(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`experince` INT(2) NOT NULL DEFAULT '4',

PRIMARY KEY (`ideml`)

);

CREATE TABLE `production` (

`idprod` INT(11) NOT NULL AUTO\_INCREMENT,

`idempl` INT(11) NOT NULL,

`id\_pr` INT(11) NOT NULL,

`percent\_flaw` INT(3) NOT NULL,

PRIMARY KEY (`idprod`)

);

CREATE TABLE `name\_product` (

`id\_pr` INT(11) NOT NULL AUTO\_INCREMENT,

`name\_product` varchar(45) NOT NULL,

`price` INT(20) NOT NULL,

PRIMARY KEY (`id\_pr`)

);

ALTER TABLE `employes` ADD CONSTRAINT `employes\_fk0` FOREIGN KEY (`ideml`) REFERENCES `user`(`iduser`);

ALTER TABLE `production` ADD CONSTRAINT `production\_fk0` FOREIGN KEY (`idempl`) REFERENCES `employes`(`ideml`);

ALTER TABLE `production` ADD CONSTRAINT `production\_fk1` FOREIGN KEY (`id\_pr`) REFERENCES `name\_product`(`id\_pr`);

# 9 Результаты тестирования системы управления проектами в компании и программной поддержки количественной оценки потенциальных рисков IT-проекта

После запуска исполняемого файла появится окно, представленное на рисунке 9.1.

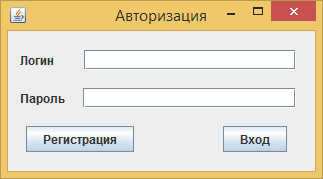


Рисунок 9.1 – Авторизация пользователя

Для того, чтобы успешно пройти авторизацию, необходимо ввести верные данные аутентификации. Система требует заполнения обеих полей. При заполнении неверными данными появится окно, представленное на рисунке 9.2.

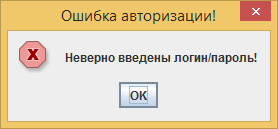


Рисунок 9.2 – Ошибка при попытке авторизации

Далее введём корректные данные, чтобы рассмотреть функционал администратора (рисунок 9.3)

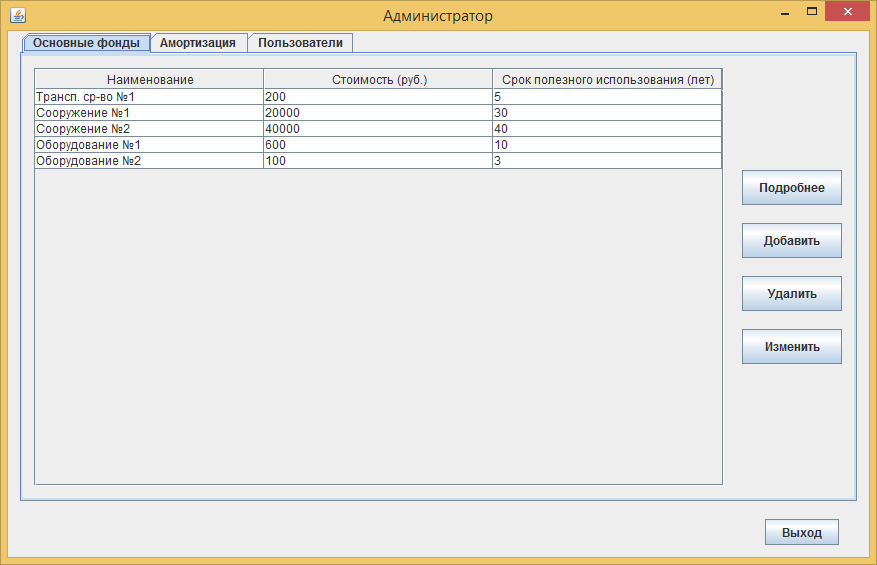


Рисунок 9.3 – Окно администратора

На рисунке представлена первая таблица «Основные фонды». Только администратор может добавлять, удалять или изменять данные. Рассмотрим добавление новой записи. При нажатии на кнопку «Добавить», появляется окно следующего вида (рисунок 9.4).

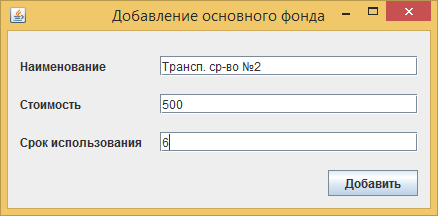


Рисунок 9.4 – Добавление новой записи

После нажатия на кнопку «Добавить», появляется сообщение об успешном добавлении (рисунок 9.5).

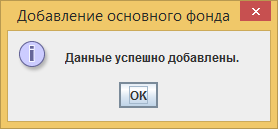


Рисунок 9.5 – Сообщение об успешном добавлении

Теперь таблица «Основные фонды» была обновлена, и можно убедиться, что данные действительно были добавлены (рисунок 9.6).

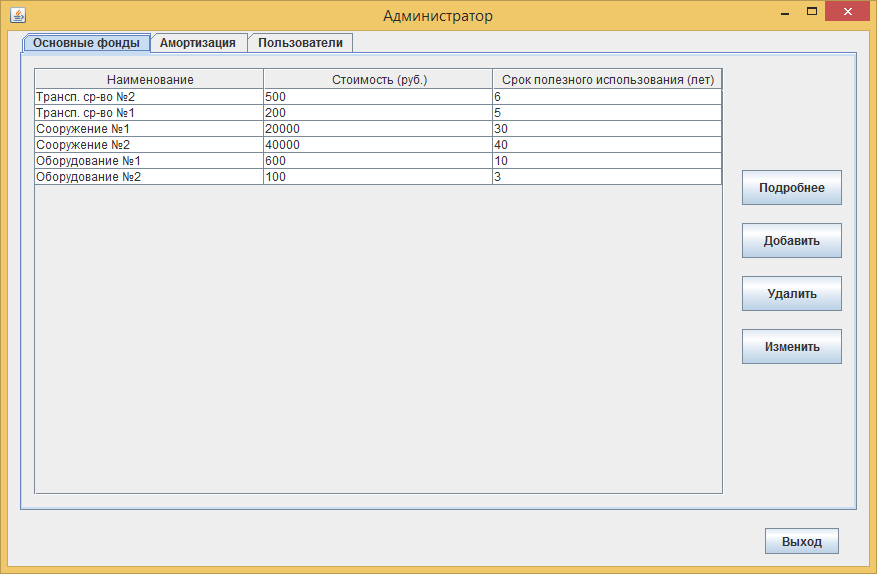


Рисунок 9.6 – Результат добавления новой записи

Рассмотрим удаление записи. Для этого следует обязательно выбрать одну запись, иначе на экране появится предупреждение (рисунок 9.7).

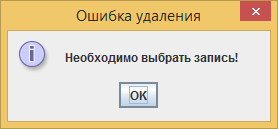


Рисунок 9.7 – Предупреждение системы

Выбрав вторую запись и нажав удалить, на экране появится сообщение об успешном удалении (рисунок 9.8).

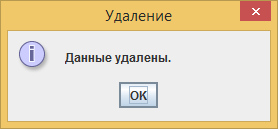


Рисунок 9.7 – Сообщение об удалении записи

Опять же таблица автоматически обновилась, в чём можно убедиться (рисунок 9.8).

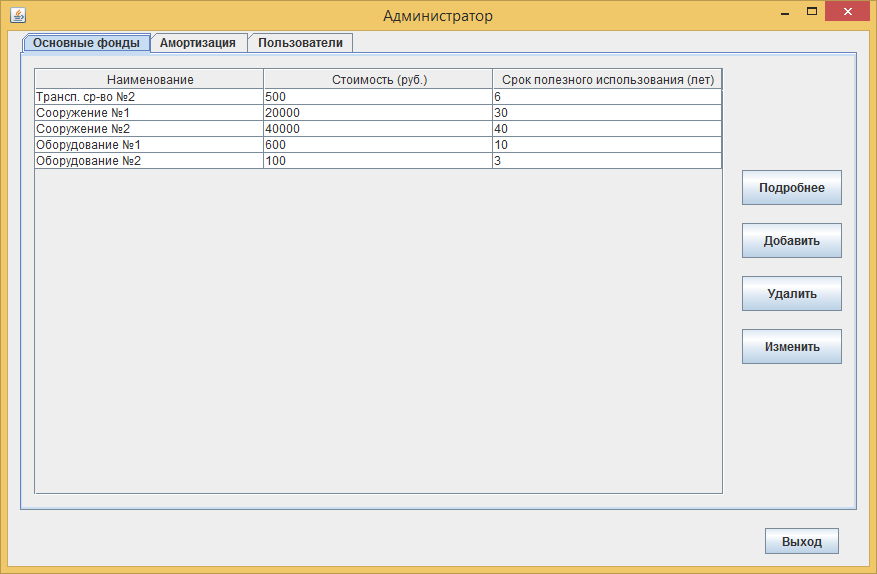


Рисунок 9.8 – Результат удаления записи

Далее перейдем на вкладку «Амортизация». На рисунке 9.9 представлена таблица с уже рассчитанными данными амортизации. Также на рисунке администратору предоставляется возможность отсортировать данные по необходимым параметрам или отфильтровать в конкретном диапазоне. Также доступен поиск данных, построение диаграммы и сознание отчёта.

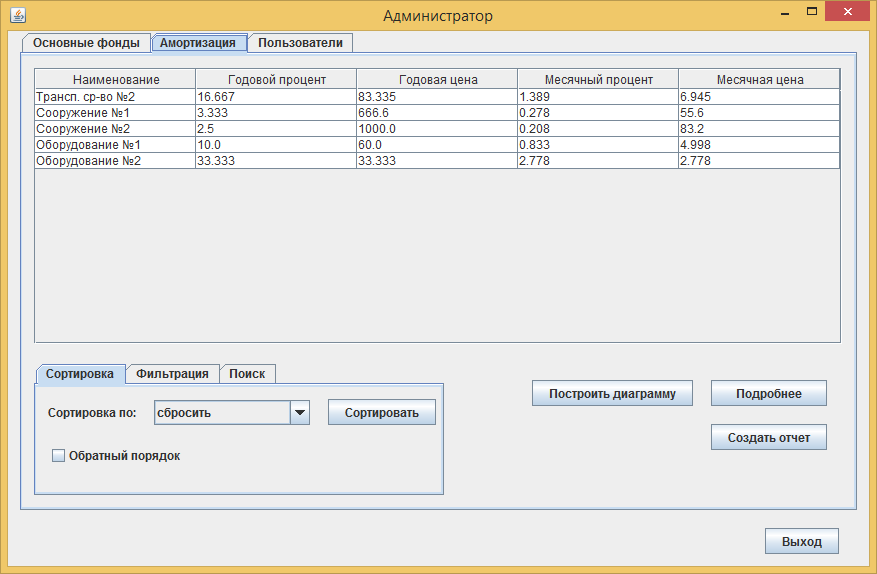


Рисунок 9.9 – Вкладка «Амортизация»

Рассмотрим функцию сортировки. Выбрав поле «Годовой процент», получим результат, представленный на рисунке 9.10.

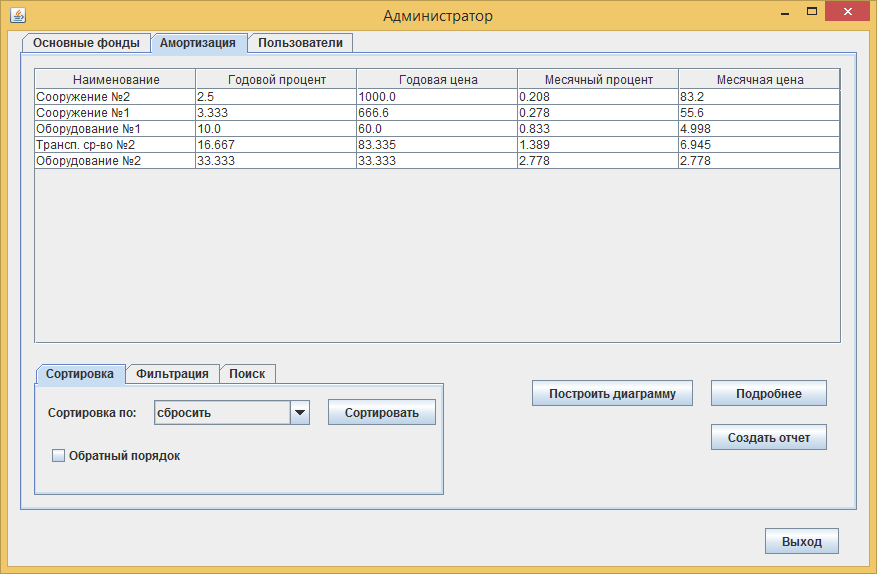


Рисунок 9.10 – Отсортированные данные

Далее сбросим сортировку и проверим функцию фильтрации. На рисунке 9.11 указан диапазон и параметр фильтрации, на рисунке 9.12 – результат.

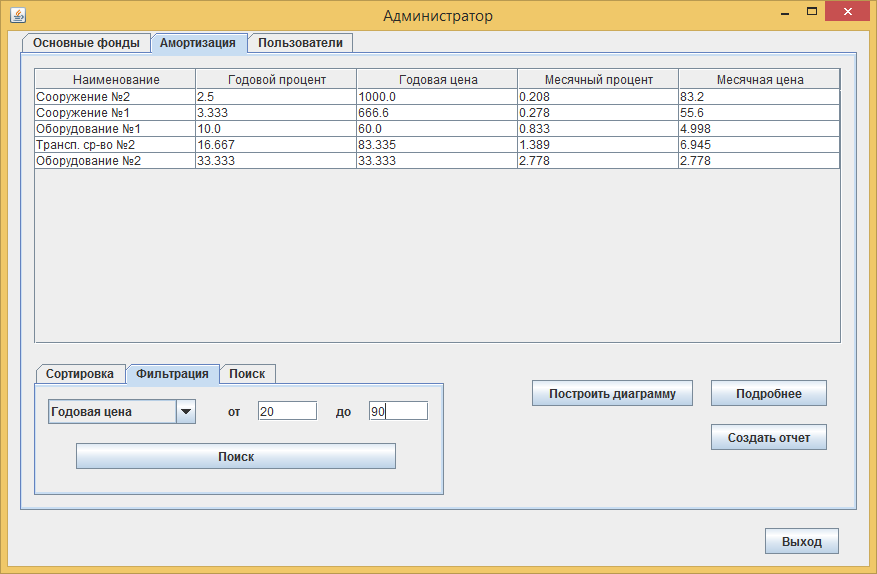


Рисунок 9.11 – Параметры фильтрации

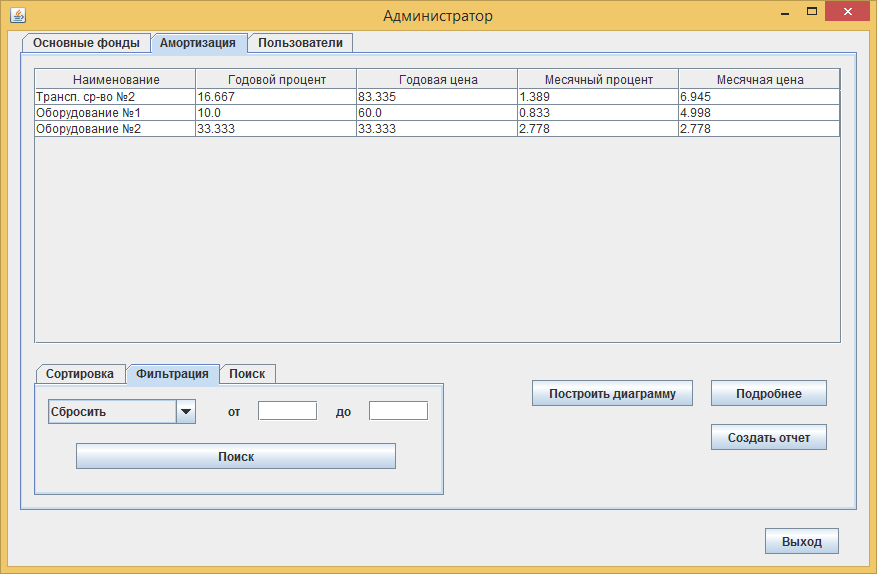


Рисунок 9.12 – Результат фильтрации

Далее рассмотрим функцию поиска. В поле введем значение «Сооружение №1», а результат поиска представлен на рисунке 9.13.

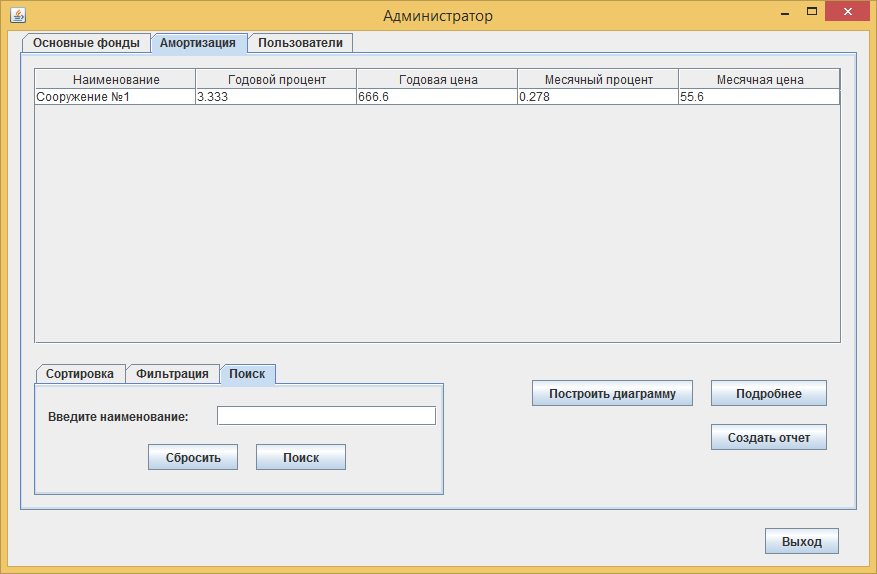


Рисунок 9.13 – Результат поиска

На панели «Амортизация» можно увидеть кнопку «Подробнее». Выбрав запись и нажав на эту кнопку, появится дополнительная информация по конкретному основному фонду из таблицы «Основные фонды» (рисунок 9.14).

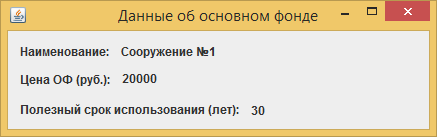


Рисунок 9.14 – Результат нажатия кнопки «Подробнее»

При выборе кнопки «Построить диаграмму» появится окно, представленное на рисунке 9.15.

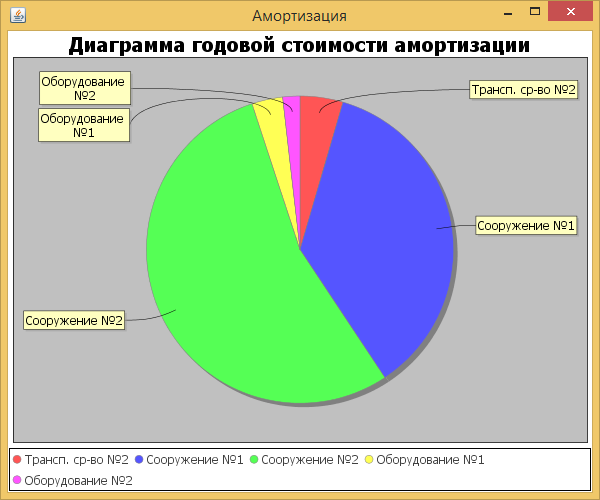


Рисунок 9.15 – Результат построения диаграммы

Далее попробуем составить отчёт. При нажатии на кнопку появится окно (рисунок 9.16), а отчёт представлен на рисунке 9.17.

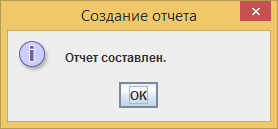


Рисунок 9.16 – Сообщение о создании отчёта

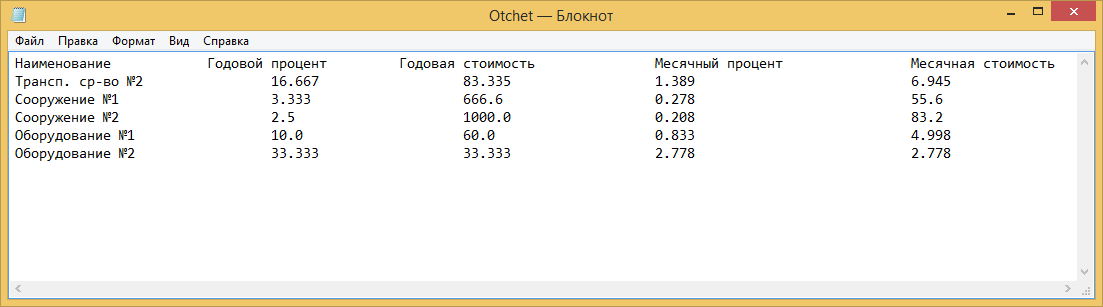


Рисунок 9.17 – Созданный отчёт

Далее перейдем на вкладку «Пользователи», где сразу выводится таблица со всеми зарегистрированными в системе пользователями (рисунок 9.18). Также присутствуют кнопки для удаления пользователей, добавления новых пользователей и изменения их статуса. Выбрав первого пользователя в списке и нажав на кнопку «Изменить статус», таблица примет вид, представленный на рисунке 9.19. Далее, выбрав первого пользователя и нажав кнопку удалить, таблица примет следующий вид (Рисунок 9.20).

Выбрав функцию «Добавить» нового сотрудника, появится окно для заполнения (рисунок 9.21). Заполнив все поля, получим следующую таблицу (Рисунок 9.22).

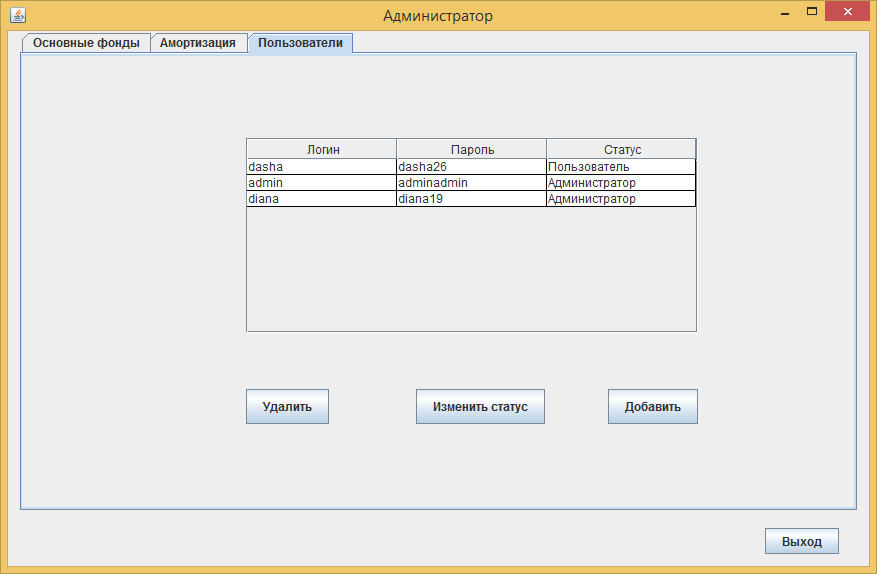


Рисунок 9.18 – Вкладка «Пользователи»

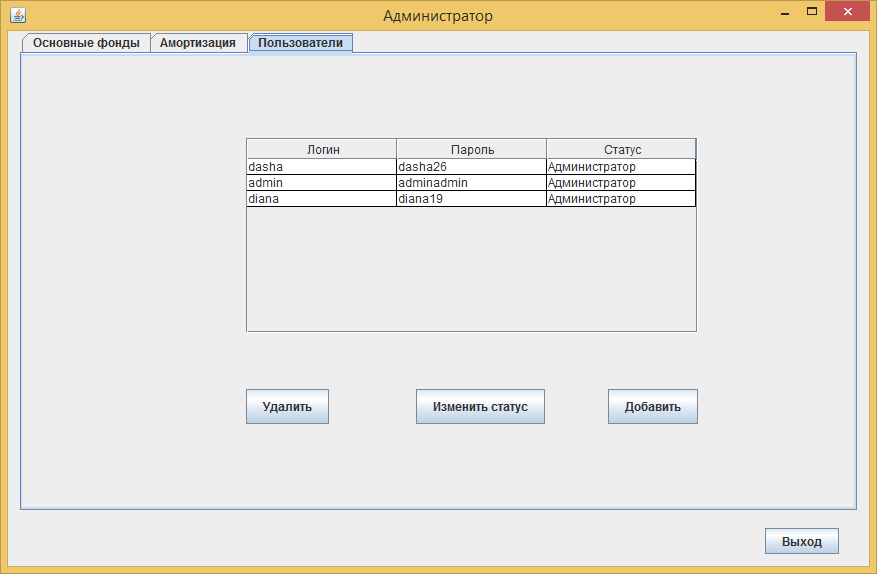


Рисунок 9.19 – Изменение статуса сотрудника

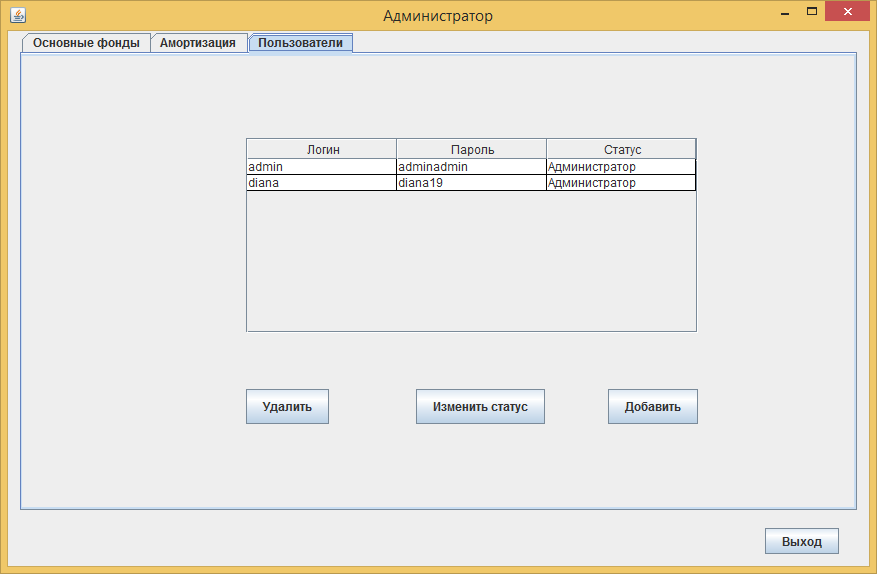


Рисунок 9.20 – Удаление сотрудника

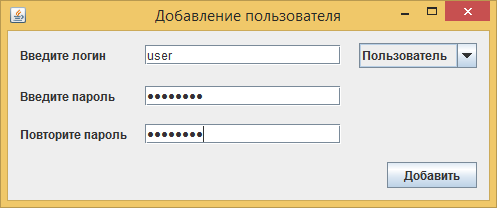


Рисунок 9.21 – Регистрация нового сотрудника

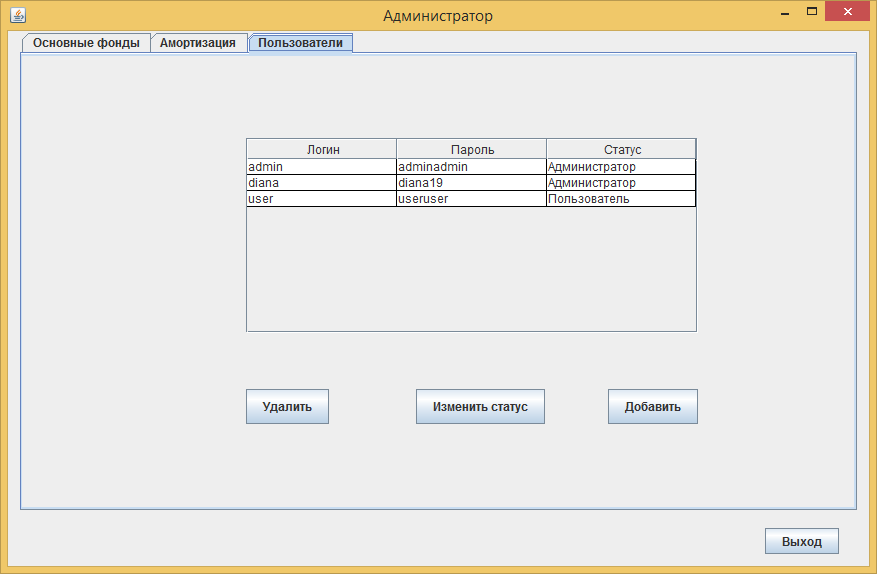


Рисунок 9.22 – Добавление сотрудника

Нажав кнопку «Выход», снова появится окно, представленное на рисунке 9.1. Теперь рассмотрим функции, доступные пользователю (Рисунок 9.23).

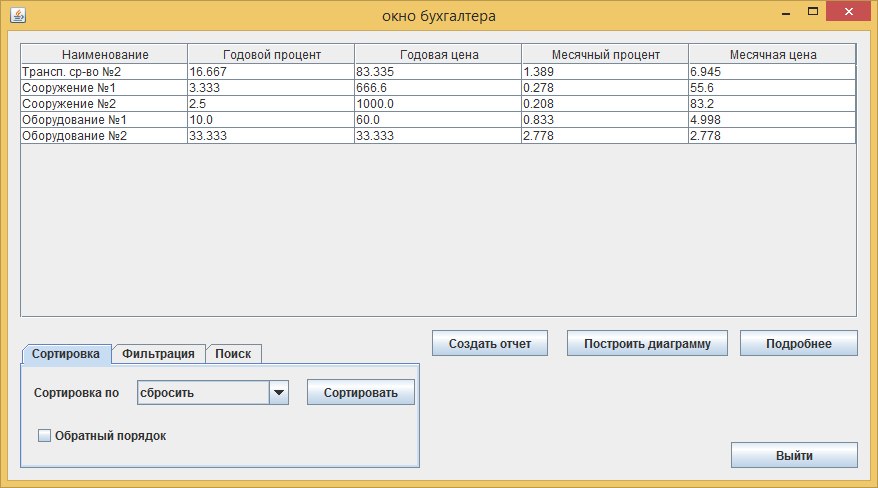


Рисунок 9.23 – Окно пользователя

На рисунке выше видно, что пользователю доступны функции, которые рассматривались ранее, только из доступа администратора.

Таким образом, мы рассмотрели основной функционал приложения и протестировали его работу.

# Заключение

Прогресс не стоит на месте, поэтому всегда актуально создавать что-то новое, например, новые писать программы, которые упростят и ускорят работу человека и у сведут ручную работ к минимуму.

Автоматизация различных сфер жизни человека является очень значимой и важной, так как сокращает время обработки различной документации и освобождает сотрудников от рутинной работы. Автоматизация учёта амортизации основных средств предприятия требует использования специального программного продукта. Так как специфика деятельности в данной сфере требует хранение больших массивов информации, а также их систематизацию, необходимо создание баз данных и систем по их управлению, чтобы обрабатывать максимальное количество информации за минимальное время.

Подводя итоги, можно сказать, что созданная программа разработана для облегчения и удобства в работе с базами данных в бухгалтерских отделах предприятий и организациях. Приложение приводит в систему данные, позволяет обеспечить централизованное хранение информации, уменьшает затраты времени на ввод данных и их удаление.

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 или требованиями.

# Список использованных источников

[1] Васильев А.Н. Java. – СПб «Питер», 2011. – 400 с.

[2] Блинов И.Н. «Java. Промышленное программирование. – Минск: «УниверсалПресс», 2007г. – 704с.

[3] Г. Буч, Д. Рамбо, А. Джекобсон. Язык UML Руководство пользователя.

[4] grandars.ru [Электронный ресурс]. – Электронные данные. – Режим доступа:http://www.grandars.ru/student/buhgalterskiy-uchet/uchet-osnovnyh-sredstv.html.

[5] ta-aspect.by [Электронный ресурс]. – Электронные данные. – Режим доступа:http://ta-aspect.by/Amortizaciya-osnovnyh-proizvodstvennyh-fondov.

[6] tic.tsu.ru [Электронный ресурс]. – Электронные данные. – Режим доступа:http://tic.tsu.ru/www/uploads/fma/2.4.2.htm.

[7] bourabai.kz [Электронный ресурс]. – Электронные данные. – Режим доступа:http://bourabai.kz/dbt/MySQL0109.htm.

# Приложение А

**(обязательное)**

**Модели представления системы (к разделу 6)**

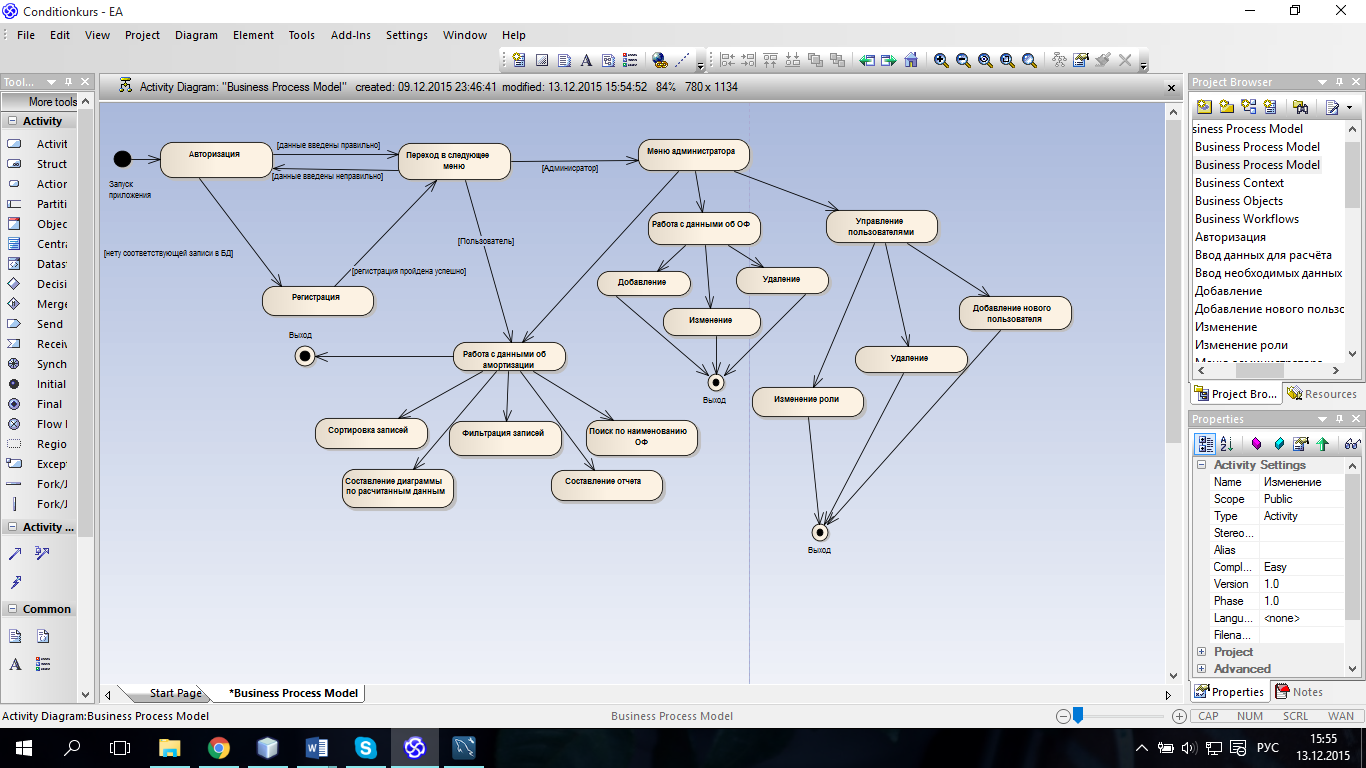


Рисунок A.1 – Диаграмма состояний (Statechart diagram)

Продолжение приложения А

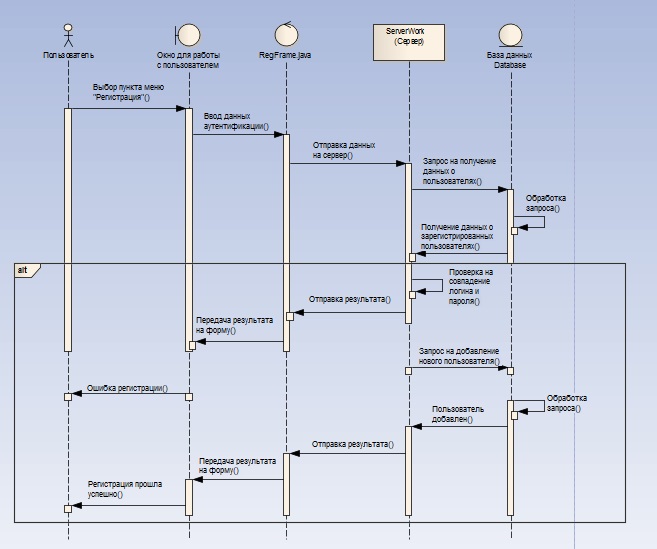


Рисунок A.2 – Диаграмма последовательностей (Sequence diagram)

Продолжение приложения А

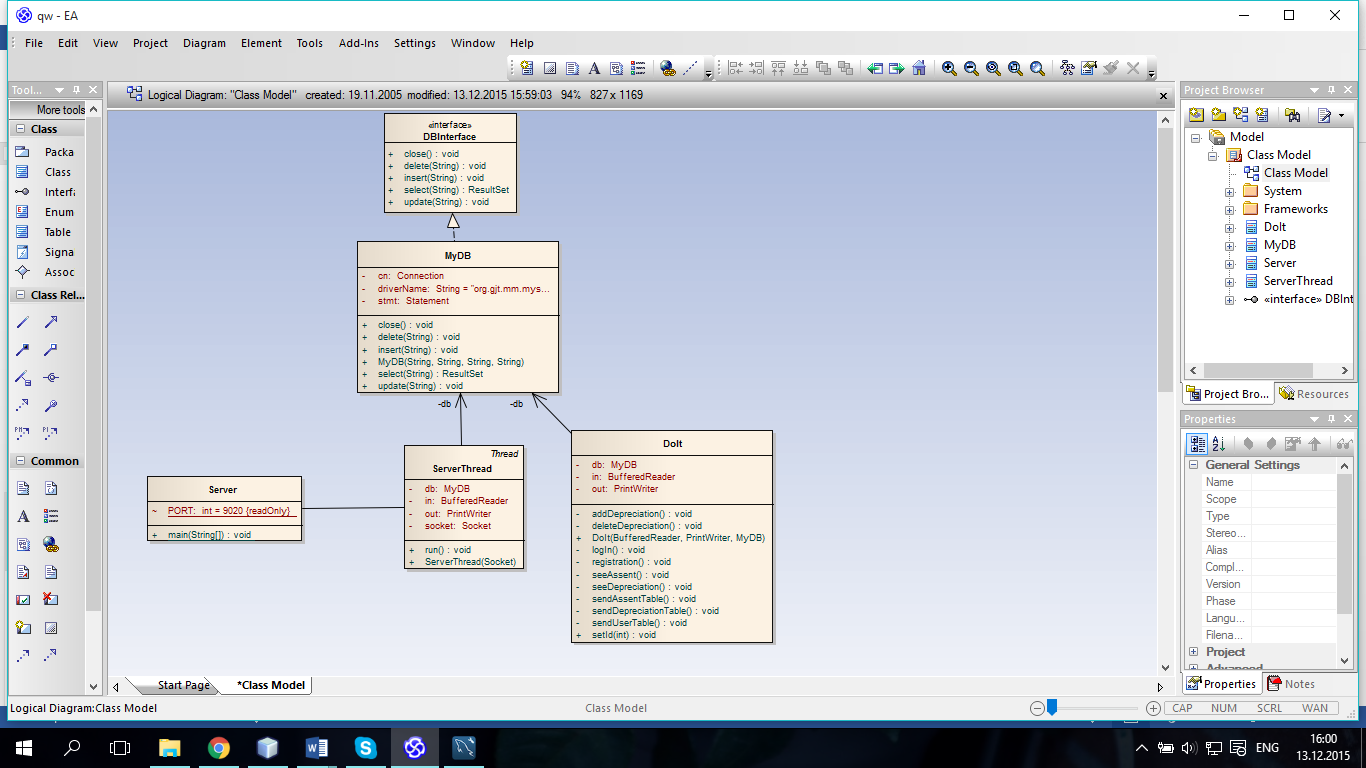


Рисунок А.3.1 – Диаграмма классов сервера

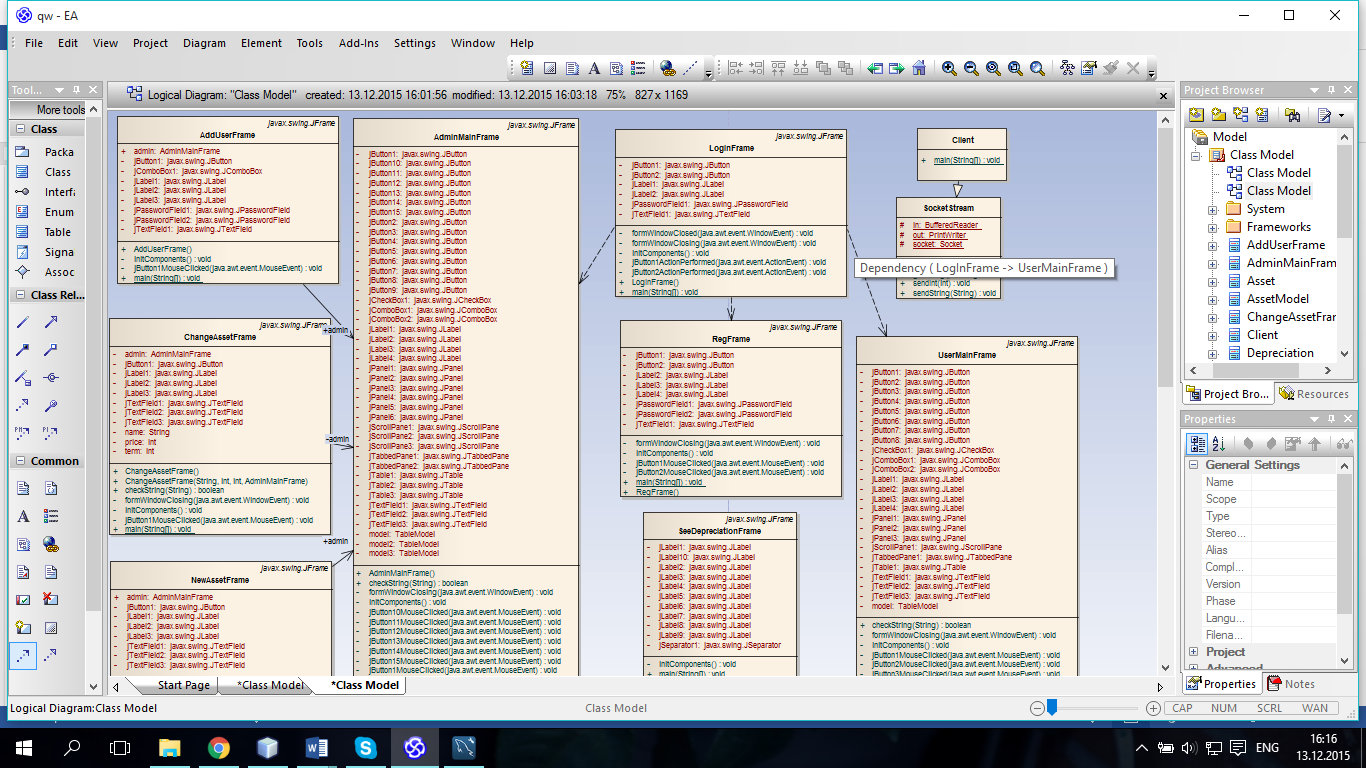


Рисунок А.3.2 – Диаграмма классов клиента

Продолжение приложения А

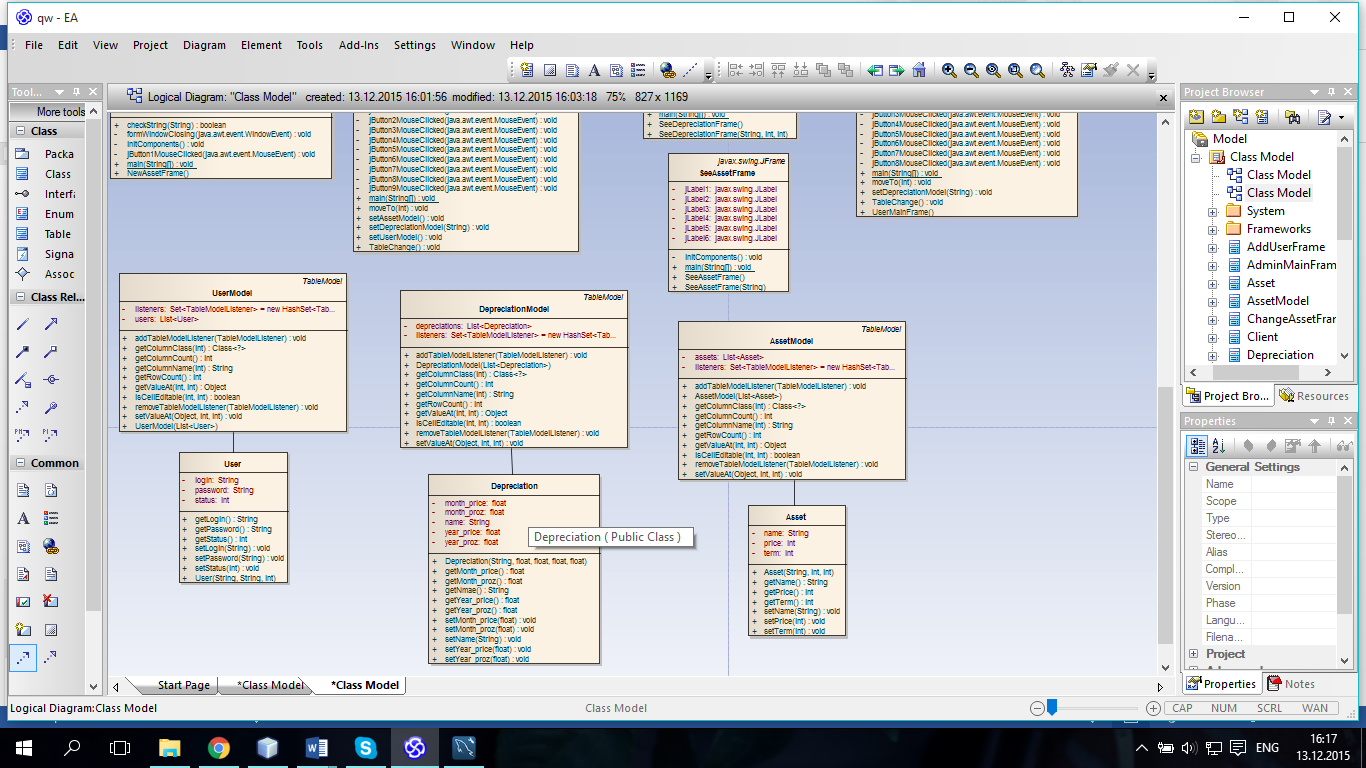


Рисунок А.3.3 – Диаграмма классов клиента

Продолжение приложения А

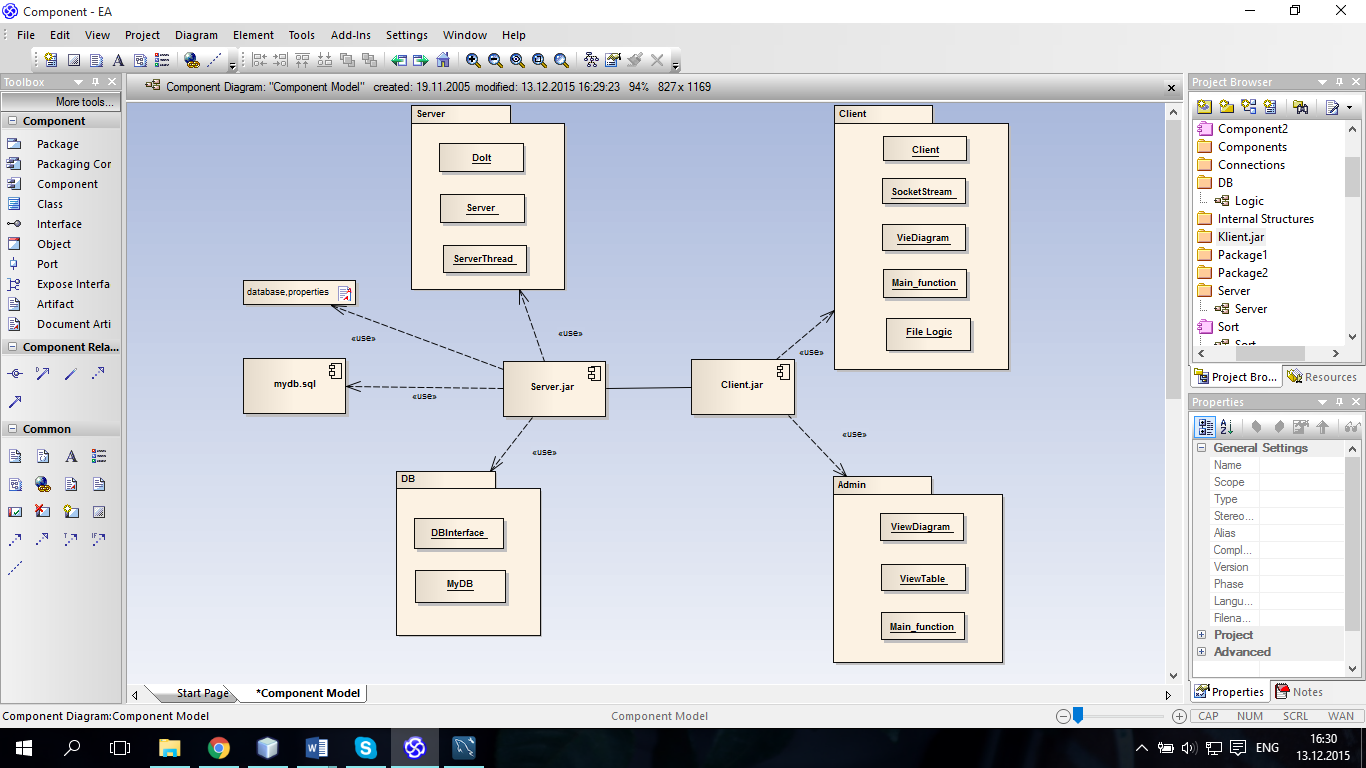


Рисунок A.4 – Диаграмма компонентов клиента (Сomponent diagram)

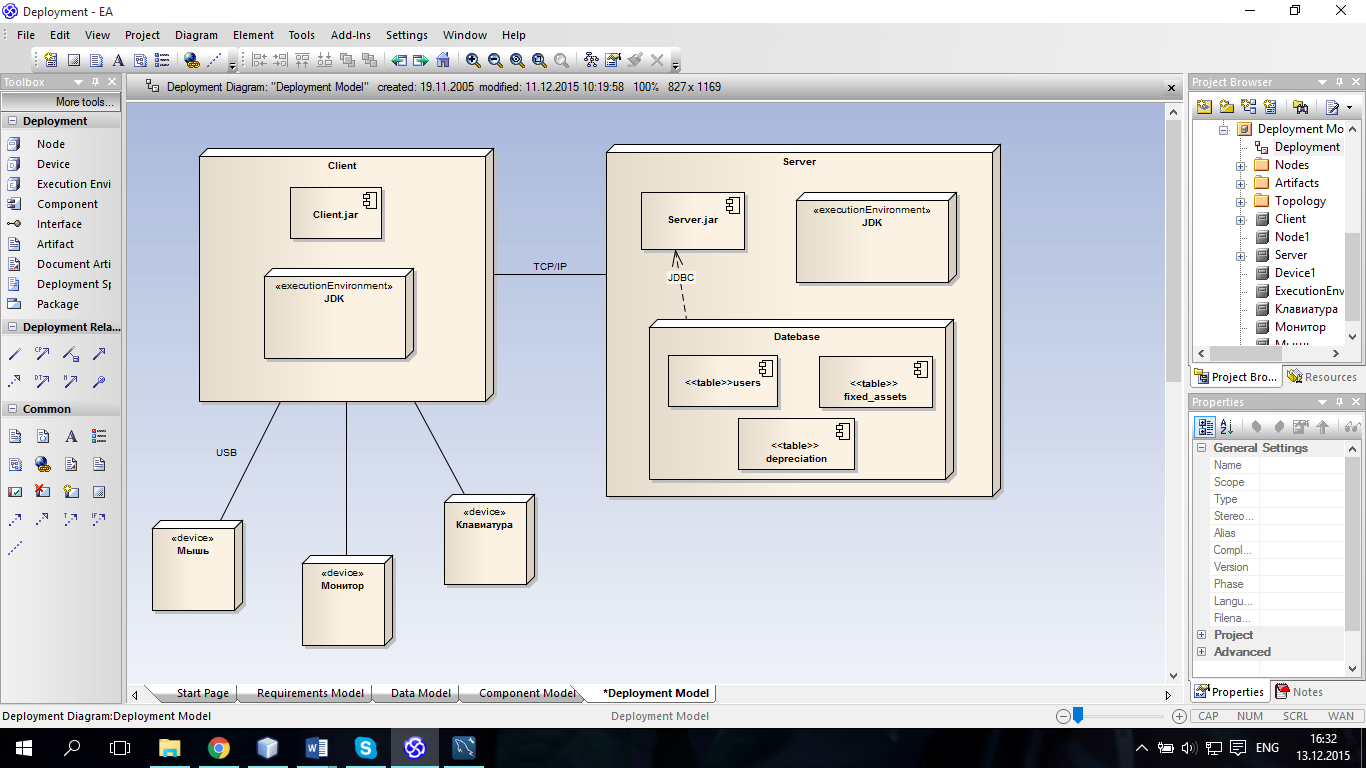


Рисунок А.5 – Диаграмма развертывания (Deployment diagram)

# Приложение Б

**(обязательное)**

**Блок-схемы программы (к разделу 7)**

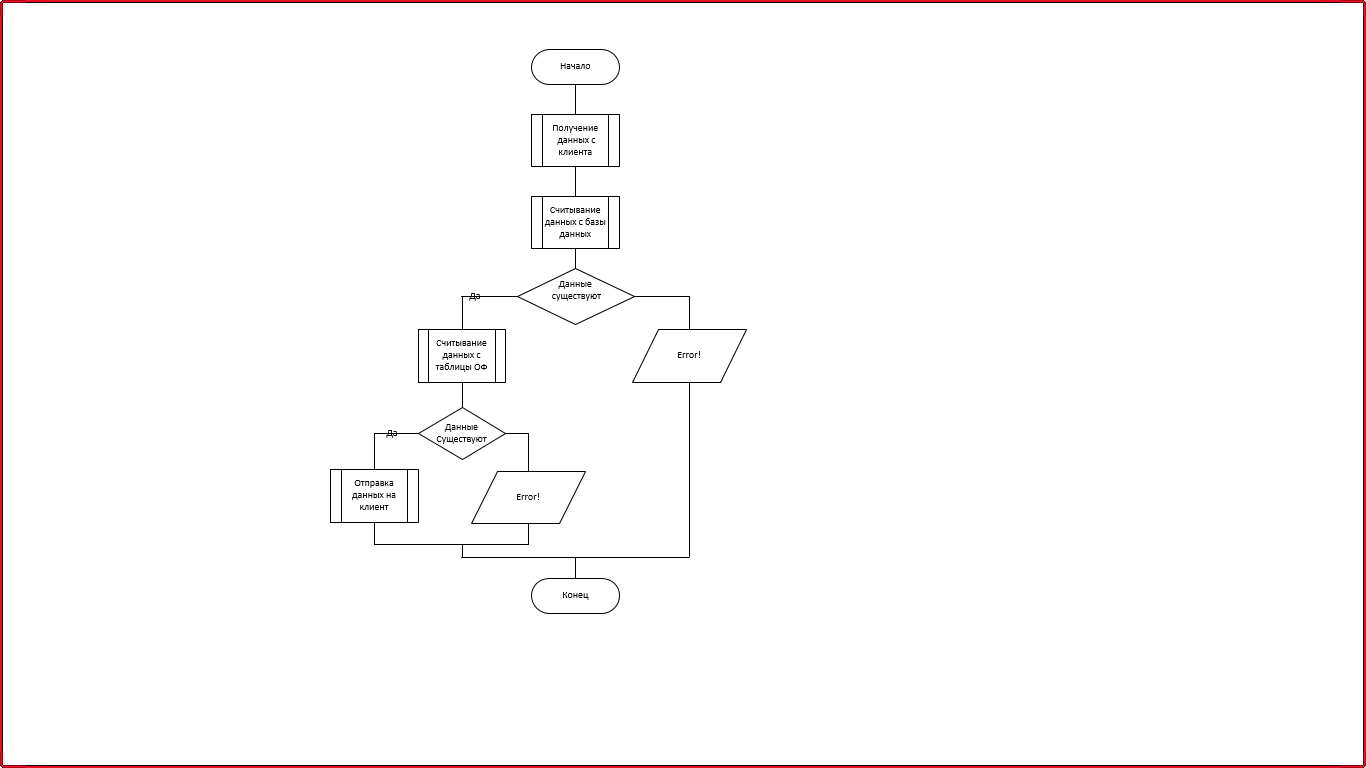


Рисунок Б.1 – Блок-схема обработка запроса на получение дополнительных данных по ОФ

Продолжение приложения Б

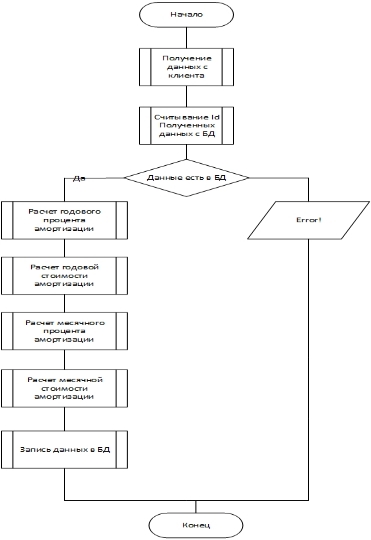
****

Рисунок Б.2 – Блок-схема расчета амортизации

# Приложение В

**(обязательное)**

**Скрипт создания БД**

CREATE TABLE `user` (

`iduser` INT(11) NOT NULL AUTO\_INCREMENT,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`status` INT(2) NOT NULL DEFAULT '0',

PRIMARY KEY (`iduser`)

);

CREATE TABLE `employes` (

`ideml` INT(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`experince` INT(2) NOT NULL DEFAULT '4',

PRIMARY KEY (`ideml`)

);

CREATE TABLE `production` (

`idprod` INT(11) NOT NULL AUTO\_INCREMENT,

`idempl` INT(11) NOT NULL,

`id\_pr` INT(11) NOT NULL,

`percent\_flaw` INT(3) NOT NULL,

PRIMARY KEY (`idprod`)

);

CREATE TABLE `name\_product` (

`id\_pr` INT(11) NOT NULL AUTO\_INCREMENT,

`name\_product` varchar(45) NOT NULL,

`price` INT(20) NOT NULL,

PRIMARY KEY (`id\_pr`)

);

ALTER TABLE `employes` ADD CONSTRAINT `employes\_fk0` FOREIGN KEY (`ideml`) REFERENCES `user`(`iduser`);

ALTER TABLE `production` ADD CONSTRAINT `production\_fk0` FOREIGN KEY (`idempl`) REFERENCES `employes`(`ideml`);

ALTER TABLE `production` ADD CONSTRAINT `production\_fk1` FOREIGN KEY (`id\_pr`) REFERENCES `name\_product`(`id\_pr`);

# 

# Приложение Г

**(обязательное)**

**Выборочный листинг кода**

**Файл Server.java**

packageServerWork;

import java.io.\*;

import java.net.\*;

importjava.sql.\*;

public class Server {

static final int PORT = 8020;

public static void main(String[] args) throws IOException,

ClassNotFoundException, SQLException {

ServerSocketservSocket = new ServerSocket(PORT);

System.out.println("Серверзапущен");

try {

while (true) {

Socket socket = servSocket.accept();

System.out.println("Был подключен новый клиент");

try {

newThreadServer(socket);

}

catch (IOExceptione) {

// Если завершится неудачей, закрывается сокет,

// в противном случае, нить закроет его:

socket.close();

}

}

}

finally {

servSocket.close();

}

}

}

**Файл ServerWork.java**

packageServerWork;

importDatabase.MyDatabase;

import java.io.\*;

import java.net.\*;

importjava.sql.\*;

importjava.util.logging.Level;

importjava.util.logging.Logger;

public class ServerWork {

privateBufferedReader in;

privatePrintWriter out;

privateMyDatabase database;

publicServerWork (BufferedReader in, PrintWriter out, MyDatabase database){

this.in = in;

this.out = out;

this.database = database;

}

public void getId (intidOperation) throws IOException, SQLException{

switch(idOperation){

case 1:

signingIn();

break;

case 2:

registrationUser();

break;

case 3:

outputAssetTable();

break;

case 4:

database.insert(in.readLine());

break;

case 5:

database.delete(in.readLine());

break;

case 6:

database.update(in.readLine());

break;

case 7:

outputUserTable();

break;

case 8:

addCapitalConsumption();

break;

case 9:

outputCapitalConsumptionTable();

break;

case 10:

deleteCapitalConsumption();

break;

case 11:

viewCapitalConsumption();

break;

case 12:

viewAsset();

break;

}

}

private void signingIn() throws IOException, SQLException{

String login = in.readLine();

String password = in.readLine();

ResultSet result;

result = database.select("select \* from users where login = '"+login+"' and password = '"+password+"'");

if(result.next()){

out.println(result.getString("status"));

}else{

out.println("error");

}

}

private void registrationUser() throws IOException, SQLException{

String login = in.readLine();

String password = in.readLine();

int status = Integer.parseInt(in.readLine());

String sqlString = "SELECT \* FROM users where login = '"+login+"'";

ResultSetrs = database.select(sqlString);

if(rs.next()){

out.println("error");

}else{

out.println("ok");

}

sqlString = "INSERT INTO users (login, password, status) VALUES ('"+login+"','"+password+"','"+status+"')";

database.insert(sqlString);

}

private void outputAssetTable() throws SQLException{

ResultSet result = database.select("SELECT \* FROM fixed\_assets");

int count = 0;

if(result.first()){

do {

count++;

} while(result.next());

}

out.println(Integer.toString(count));

result.first();

for(inti = 0; i< count; i++){

out.println(result.getString("name"));

out.println(Integer.toString(result.getInt("price")));

out.println(Integer.toString(result.getInt("term\_of\_use")));

result.next();

}

}

private void outputUserTable() throws SQLException{

ResultSet result = database.select("SELECT \* FROM users");

int count = 0;

if(result.first()){

do{

count++;

}while(result.next());

}

out.println(Integer.toString(count));

result.first();

for(inti = 0; i< count; i++){

out.println(result.getString("login"));

out.println(result.getString("password"));

out.println(Integer.toString(result.getInt("status")));

result.next();

}

}

private void addCapitalConsumption() throws IOException, SQLException{

String name = in.readLine();

int price = Integer.parseInt(in.readLine());

int term = Integer.parseInt(in.readLine());

String sqlString = "SELECT id FROM fixed\_assets WHERE `name`='"+name+"'"

+ " AND `price`='"+price+"' AND `term\_of\_use`='"+term+"'";

ResultSet result = database.select(sqlString);

int id = 0;

if(result.next()){

id = result.getInt("id");

floatyear\_percent = ((float)100)/term;

year\_percent = ((float)Math.round(year\_percent \* 1000) ) / 1000;

floatyear\_price = price/100\*year\_percent;

year\_price = ((float)Math.round(year\_price \* 1000) ) / 1000;

floatmonth\_percent = year\_percent/12;

month\_percent = ((float)Math.round(month\_percent \* 1000) ) / 1000;

floatmonth\_price = price/100\*month\_percent;

month\_price = ((float)Math.round(month\_price \* 1000) ) / 1000;

sqlString = "INSERT INTO depreciation (name, year\_proz, year\_price, month\_proz,"

+ " month\_price, id\_fixed\_assets) VALUES ('"+name+"','"+year\_percent+"',"

+ "'"+year\_price+"','"+month\_percent+"','"+month\_price+"','"+id+"')";

database.insert(sqlString);

}

}

private void outputCapitalConsumptionTable() throws SQLException, IOException{

ResultSet result = database.select(in.readLine());

int count = 0;

if(result.first()){

do{

count++;

}while(result.next());

}

out.println(Integer.toString(count));

result.first();

for(inti = 0; i< count; i++){

out.println(result.getString("name"));

out.println(Double.toString((double)result.getFloat("year\_proz")));

out.println(Double.toString((double)result.getFloat("year\_price")));

out.println(Double.toString((double)result.getFloat("month\_proz")));

out.println(Double.toString((double)result.getFloat("month\_price")));

result.next();

}

}

private void deleteCapitalConsumption() throws IOException, SQLException{

ResultSet result = database.select(in.readLine());

if(result.next()){

int id = result.getInt("id");

String sqlString = "DELETE FROM depreciation WHERE `id\_fixed\_assets`='"+id+"'";

database.delete(sqlString);

}

return;

}

private void viewCapitalConsumption() throws IOException, SQLException{

ResultSet result = database.select(in.readLine());

if(result.next()){

int id = result.getInt("id");

String sqlString = "SELECT \* FROM depreciation WHERE `id\_fixed\_assets`='"+id+"'";

result = database.select(sqlString);

if(result.next()){

out.println(result.getString("name"));

out.println(Double.toString((double)result.getFloat("year\_proz")));

out.println(Double.toString((double)result.getFloat("year\_price")));

out.println(Double.toString((double)result.getFloat("month\_proz")));

out.println(Double.toString((double)result.getFloat("month\_price")));

}

}

}

private void viewAsset() throws IOException, SQLException{

ResultSet result = database.select(in.readLine());

if(result.next()){

out.println(result.getString("name"));

out.println(Integer.toString((int)result.getInt("price")));

out.println(Integer.toString((int)result.getInt("term\_of\_use")));

}

}

}