Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра экономической информатики

Дисциплина: Программирование сетевых приложений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ОЦЕНКИ ФИНАНСОВЫХ РИСКОВ КОМПАНИИ**

Выполнила: студент гр.710101

Самчук А.В.

Проверил: Петрович Н.О.

Минск 2019

**СОДЕРЖАНИЕ**

Введение ……………………………………………………………..……………3

1 Описание предметной области ………………………………………………5

* 1. Финансовые риски предприятий: сущность, виды, методы расчёта....5
  2. Определение требований к системе …………………………………....8

2 Постановка задачи и обзор методов её решения ………………………...…9

2.1 Постановка задачи и обзор методов её решения ……………………..9

2.1.1 JAVAFX …………………………………………………………......10

3 Функциональное моделирование на основе стандарта IDEF0 ……….…..11

4 Информационная модель системы и её описание ……………….………..15

5 Модели представления системы и их описание …………………..………17

5.1 Диаграмма вариантов использования …………………………...…….17

5.2 Диаграмма состояний …………………………………………..………17

5.3 Диаграмма классов………………………………………….…………..17

5.4 Диаграмма последовательностей …………………………….………..18

5.5 Диаграмма развертывания ……………………………………….….…18

6 Описание алгоритмов, реализующих бизнес-логику системы .........……19

6.1 Алгоритм проверки того, заблокирован ли пользователь …………...19

7 Руководство пользователя …………………………………………...…….20

Заключение …………………………………………………………………..…..27

Список используемых ресурсов ……………………………………………..…28

Приложение А(обязательное) Диаграмма вариантов использования …….…29

Приложение Б(обязательное) Диаграмма состояний ………………………....30

Приложение В(обязательное) Диаграмма классов …………………………....31

Приложение Г(обязательное) Диаграмма последовательностей ...………..…37

Приложение Д(обязательное) Диаграмма развертывания …………….…..….38

Приложение Е(обязательное) Блок-схема …….………………….…..……….39

Приложение Ж(обязательное) Листинг кода …….…………………...……….40

Приложение З(обязательное) Скрипт генерации базы данных ……….……..53

**ВВЕДЕНИЕ**

Целью предпринимательства является получение максимальных доходов при минимальных затратах капитала в условиях конкурентной борьбы. Реализация указанной цели требует соизмерения размеров вложенного в производственно-торговую деятельность капитала с финансовыми результатами этой деятельности.

Вместе с тем, при осуществлении любого вида хозяйственной деятельности объективно существует опасность - «риск» потерь, объем которых обусловлен спецификой конкретного бизнеса. Риск-это вероятность возникновения потерь, убытков, не поступлений планируемых доходов, прибыли. В случае наступления подобного события возможны три принципиально разных результата:

* убыток;
* прибыль;
* нулевой результат.

Безусловно, риском можно управлять, т. е., принимать меры к прогнозированию наступления рискового события и далее разрабатывать комплекс мероприятий, позволяющих снизить степень риска либо уменьшить его отрицательные последствия.

Особенностью финансового риска является вероятность наступления ущерба в результате проведения каких-либо операций в финансово-кредитной и биржевой сферах, совершения операций с фондовыми ценными бумагами, т. е., риска, который вытекает из природы этих операций.

Для выживания в современных условиях, и даже в период завершения международного финансового кризиса компаниям, компании следует производить оценку своих возможностей для обеспечения существования на рынке. Основной структурой в компании, которую следует оценивать со всех сторон, является оценка финансовых рисков компании. В современное время оценка финансовых рисков компании является достаточно актуальной темой.

Актуальность темы сводится к следующему, что оценка финансовых рисков позволит компании устоять на рынке своих услуг или товаров или работ. Также актуальность заявленной темы выраженности в стабильности компании, возможном регулировании своих потерь в финансовой части, а и в последующем оценка финансовых рисков позволит эффективно вести работу по управлению персоналом в своей компании.

Оценка финансового состояния может быть выполнена с различной степенью детализации в зависимости от цели анализа, имеющейся информации, программного, технического и кадрового обеспечения. Наиболее целесообразным является выделение процедур экспресс-анализа и углубленного анализа финансового состояния.

Таким образом, цель данного курсового проекта заключается в повышении эффективности работы специалистов по оценке финансовых рисков предприятия путем упрощения и автоматизации процесса по расчету финансовых рисков и сопутствующих показателей.

Поставленная цель будет достигнута благодаря решению следующих задач:

* провести анализ предметной области с учетом целей работы;
* моделирование бизнес-задач и выбор способов их решения;
* создание и описание программного модуля расчета финансовых рисков;
* организация работы с базой данных;
* разработка автоматизированной системы с пользовательским интерфейсом;

1. **ХАРАКТЕРИСТИКА ФИНАНСОВЫХ РИСКОВ И МЕТОДОВ ИХ ОЦЕНКИ.**

**ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ**

* 1. **Финансовые риски предприятий: сущность, виды, методы расчета**

Финансовый риск предприятия – это вероятность возникновения неблагоприятных финансовых последствий в форме потери дохода или капитала при неопределенности условий осуществления его финансовой деятельности.

Под финансовым понимается риск, возникающий при осуществлении финансового предпринимательства или финансовых сделок, исходя из того, что в финансовом предпринимательстве в роли товара выступают либо ценные бумаги, либо денежные средства в рублях или валюте. Финансовые риски можно разделить на:

* валютный риск;
* кредитный риск;
* инвестиционный риск.

Валютный риск **-** это вероятность финансовых потерь в результате изменения курса валют, которое может произойти в период между заключением контракта и фактическим проведением расчетов по нему. Валютный риск включает в себя три разновидности:

* экономический риск;
* риск перевода;
* риск сделок.

Кредитный риск связан с возможностью невыполнения предпринимательской фирмой своих финансовых обязательств перед инвестором в результате использования для финансирования деятельности фирмы внешнего займа.

Инвестиционный риск связан со спецификой вложения денежных средств предприятия в различные инвестиционные проекты. Его основные виды:

* капитальный;
* селективный;
* процентный;
* страновой;
* операционный;
* временной;
* риск законодательных изменений;
* риск ликвидности;
* инфляционный;

Методический инструментарий оценки уровня финансового риска является наиболее обширным, так как включает в себя разнообразные экономико-статистические, экспертные, аналоговые методы осуществления такой оценки.

Существуют следующие методы оценки риска:

Экономико-статистические методы составляют основу проведения оценки уровня финансового риска. К числу основных расчетных показателей такой оценки относятся:

а) Уровень финансового риска. В практике использования этого алгоритма размера возможных финансовых потерь выражается обычно абсолютной суммой, а вероятность возникновения финансового риска — одним из коэффициентов измерения этой вероятности (коэффициентом вариации,

бета-коэффициентом и др.)

б) Дисперсия. Она характеризует степень колеблемости изучаемого показателя (в данном случае — ожидаемого дохода от осуществления финансовой операции) по отношению к его средней величине.

в) *Среднеквадратическое (стандартное) отклонение.* Этот показатель является одним из наиболее распространенных при оценке уровня индивидуального финансового риска, так же как и дисперсия определяющий степерь колеблемости и построенный на её основе.

г) *Коэффициент вариации*. Он позволяет определить уровень риска, если показатели среднего ожидаемого дохода от осуществления финансовых операций различаются между собой.

Экспертные методы оценки уровня финансового риска применяются в том случае, если на предприятии отсутствуют необходимые информативные данные для осуществления расчетов экономико-статистическими методами. Эти методы базируются на опросе квалифицированных специалистов с последующей математической обработкой результатов этого опроса.

В процессе экспертной оценки каждому эксперту предлагается оценить уровень возможного риска, основываясь на определенной балльной шкале.

Аналоговые методы оценки уровня финансового риска позволяют определить уровень рисков по отдельным наиболее массовым финансовым операциям предприятия. При этом для сравнения может быть использован как собственный, так и внешний опыт осуществления таких финансовых операций.

Уровень финансового риска*.* Он характеризует общий алгоритм оценки этого уровня, представленный следующей формулой:

,

Где УР – уровень соответствующего финансового риска;

ВР – вероятность возникновения данного финансового риска;

РП – размер возможных финансовых потерь при реализации данного риска.

Дисперсия**.** Расчет дисперсии осуществляется по следующей формуле:

https://studfile.net/html/2706/966/html_6HSn0Dmppd.SxBi/img-aT3xWW.jpg

где https://studfile.net/html/2706/966/html_6HSn0Dmppd.SxBi/img-UMml8v.png— дисперсия; Ri— конкретное значение возможных вариантов

ожидаемого дохода по рассматриваемой финансовой операции;

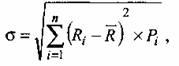
– среднее ожидаемое значение дохода по рассматриваемой финансовой операции;



PI – возможная частота (вероятность) получения отдельных вариантов ожидаемого дохода по финансовой операции;

n – число наблюдений.

Среднеквадратическое (стандартное) отклонение рассчитывается по следующей формуле:



где https://studfile.net/html/2706/966/html_6HSn0Dmppd.SxBi/img-9ex4al.png– Среднеквадратическое (стандартное) отклонение;

RI – конкретное значение возможных вариантов ожидаемого дохода по рассматриваемой финансовой операции;

– среднее ожидаемое значение дохода по рассматриваемой финансовой операции;



PI – возможная частота (вероятность) получения отдельных вариантов ожидаемого дохода по финансовой операции;

n – число наблюдений.

Коэффициент вариации*.* Расчет коэффициента вариации осуществляется по следующей формуле:

https://studfile.net/html/2706/966/html_6HSn0Dmppd.SxBi/img-hKA5ob.jpg

где CV – коэффициент вариации;

– среднеквадратическое (стандартное) отклонение;



– среднее ожидаемое значение дохода по рассматриваемой финансовой операции.



* 1. **Определение требований к системе**

В данном случае система – это приложение в архитектуре клиент-сервер с многопоточным сервером с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

Для решения конкретных поставленных задач программный продукт должен отвечать ряду технологических требований:

* система управления базами данных(СУБД): Sybase SQL 11.0+, MS SQL Server 2008 R2+, MySQL 5.5+, PostgreSQL 9.0+, Java DB 10.x+;
* операционная система Windows 7 и выше;
* интерфейс программы и данные должны быть только на русском языке;
* приложение должно запускаться без использования интегрированных средств разработки;
* должна быть выполнено тестирование каждой бизнес-функции приложения;
* разработка и использование собственной иерархии классов (не менее 5), расширение базовых классов, предоставляемых JDK;
* реализация не менее 2-х паттернов проектирования на свой выбор (по желанию).
* использовать сокрытие данных (инкапсуляция), перегрузку методов, переопределение методов, сериализацию, абстрактные типы данных (интерфейсы, абстрактные классы), статические методы, обработку исключительных ситуаций;
* в консольном окне серверной части должна отображаться текущая статистика подключений и параметры запуска сервера;
* предусмотреть механизм авторизации пользователей (роли Администратор и Пользователь);
* подписи на всех элементах схем и диаграмм, за исключением названий классов (сущностей), переменных, методов и атрибутов, должны быть выполнены на русском языке русскими буквами;
* необходим механизм регистрации с авторизацией пользователей для допуска функциям;
* требуется понятный, удобный интерфейс ко всем функциям;

1. **ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЁ РЕШЕНИЯ**
   1. **Постановка задачи и обзор методов её решения**

Задача данного курсового проекта – это создание функционирующего приложения в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java. Серверное приложение может быть реализовано в виде консольного приложения или GUI-приложения. Клиентское приложение: оконное приложение с использованием средств создания пользовательского интерфейса (JAVAFX,Scene Builder).

Для успешной реализации системы необходимо, в первую очередь, выделить основные задачи, которые будет решать система «Оценки финансовых рисков компании», а также те задачи, которые необходимо выполнить для правильной работы системы.

Основными задачами являются следующие:

* 1. Разработка базы данных для MySQL 5.5+.
  2. Разработка и использование собственной иерархии классов, расширение базовых классов, предоставляемых JDK.
  3. Реализация не менее 2-х паттернов проектирования.
  4. Использовать сокрытие данных (инкапсуляция), перегрузку методов, переопределение методов, абстрактные типы данных (интерфейсы, абстрактные классы), статические методы, обработку исключительных ситуаций.
  5. В разрабатываемом приложении обеспечить добавление, редактирование и удаление записей из базы данных, сохранение табличных результатов в файле (создание текстового отчета), предоставление пользователю аналитической информации (графики, диаграммы).
  6. Предусмотреть возможность сохранять информацию в любой момент на сервере и загружать ранее сохраненные данные.
  7. В консольном окне серверной части должна отображаться текущая статистика подключений и параметры запуска сервера.
  8. Предусмотреть механизм авторизации пользователей (роли Администратор и Пользователь).

Система должна предоставлять удобный интерфейс для конечного пользователя.

Программа дает возможность финансисту улучшить производительность труда, посредством быстрого доступа ко всей интересующей информации.

Рассмотрим некоторые функции, реализующие бизнес-логику:

* Регистрация пользователей.

Пользователь должен заполнить форму регистрации и нажать кнопку «Зарегистрироваться». Если с данными пользователя все в порядке, то он будет зарегистрирован и сможет начать свою работу.

* Авторизация.

Пользователь вводит свой логин и пароль. Если введённый логин есть в базе данных, то дальше проверяем, совпадает ли пароль, введённый пользователем с паролем в базе данных. Если запись в БД не найдена или введён не верный пароль, то будет выведено сообщение об ошибке. Иначе выведется сообщение об успешном входе и в зависимости от роли пользователя, у него будут определённые привилегии.

– Просмотр списка пользователей.

Формируется SQL-запрос, и выводится информация о пользователях, их логин и роль в виде таблицы.

– Просмотр финансовых операций.

Формируется SQL-запрос, и выводится информация, о финансовых операциях выбранного предприятия, их риску и показателям.

– Визуализация информации посредством круговой диаграммы.

Строится диаграмма, показывающая соотношений показателей финансовой операции.

– Расчет показателя финансовой операции.

Необходимо выбрать операцию из таблицы и нажать соответствующую кнопку для расчета показателя.

* + 1. **JAVAFX**

JAVAFX – платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing.

1. **ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0**

Основным процессом предметной области является оценка финансового риска компании. На рисунке 3.1 представлена контекстная диаграмма «Рассчитать финансовый риск». Контекстная диаграмма состоит из одного блока, описывающего функцию верхнего уровня, её входы, выходы и механизмами управления.

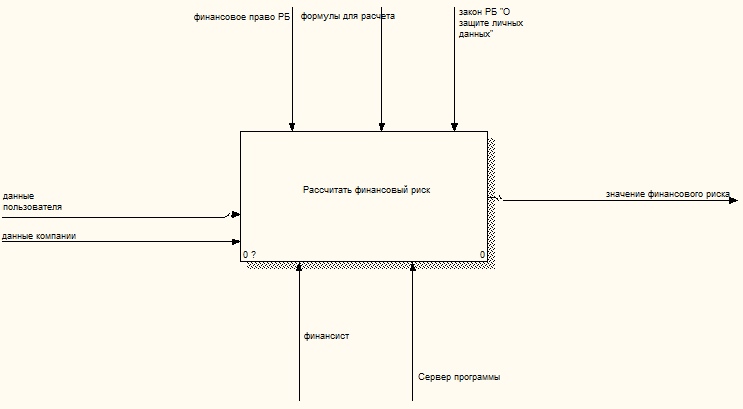


Рисунок 3.1 – Контекстная диаграмма «Рассчитать финансовый риск»

В качестве входных данных выступают данные пользователя и данные компании, для финансовой операции которой и будет рассчитан финансовый риск. Механизмами управления являются финансист и сервер программы. Выходными данными будет численное значение уровня риска.

Проектирование проводится путем декомпозиции – разделения рассматриваемой функции на подфункции.

В ходе декомпозиции блока «Рассчитать финансовый риск» он был разделен на 4 подблока:

* Авторизация;
* Выбрать компанию;
* Получить финансовую операцию;
* Рассчитать риски;

Результат декомпозиции представлен на рисунке 3.2.

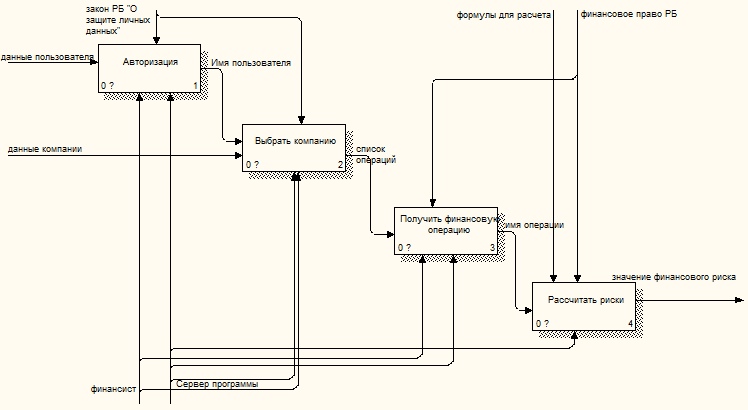


Рисунок 3.2 – процесс рассчёта финансового риска

Также была произведена декомпозиция блока «Авторизация», в результате чего он был разделен на 4 подблока:

* ввести логин;
* ввести пароль;
* проверить наличие учетной записи в системе;
* получить имя пользователя.

Результат декомпозиции представлен на рисунке 3.3.

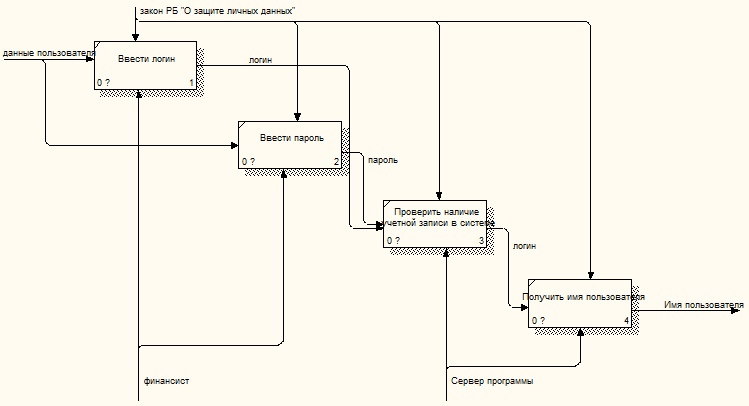


Рисунок 3.3 – результат декомпозиции блока «Авторизация»

Была проведена декомпозиция блока «Выбрать компанию», в результате чего он был разбит на 3 подблока:

* вывод таблицы компаний;
* выбор компании из таблицы;
* нажатие кнопки просмотра операций.

Результат декомпозиции приведен на рисунке 3.4.

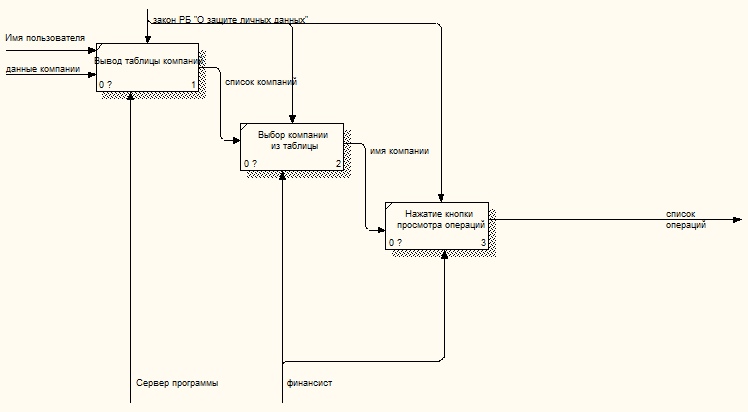


Рисунок 3.4 – результат декомпозиции блока «Выбрать компанию»

Также была проведена декомпозиция процесса «Получить финансовую операцию», в результате чего процесс был разделен на 2 подпроцесса:

* получить данные финансовых операций;
* выбрать операцию из таблицы.

Результат декомпозиции приведён на рисунке 3.5.

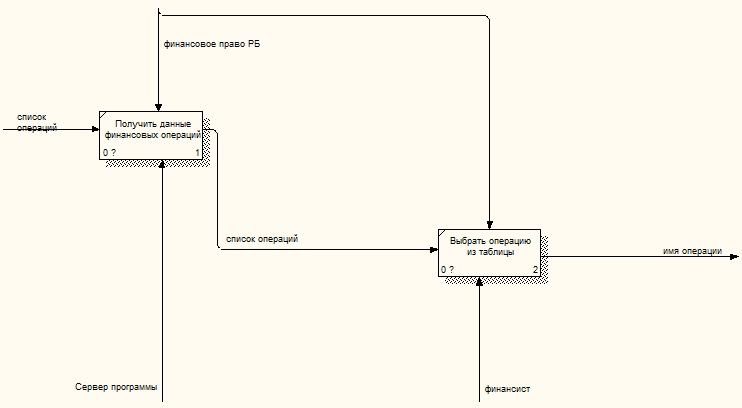


Рисунок 3.5 – результат декомпозиции процесса «Получить финансовую операцию»

Также была проведена декомпозиции блока «Рассчитать риски», в результате чего тот был разбит на 2 подблока:

* получить рассчетные данные из бд;
* рассчёт риска.

Результат декомпозиции представлен на рисунке 3.6.

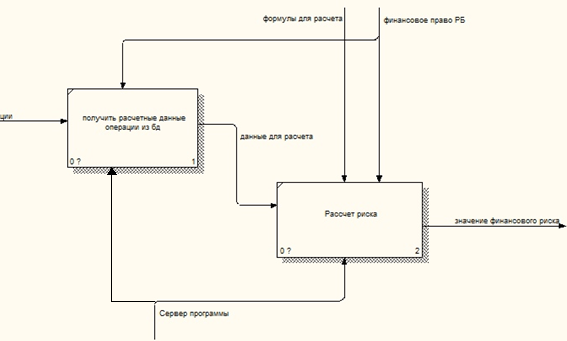


Рисунок 3.6 – результат декомпозиции процесса «Рассчитать риски»

1. **ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ**

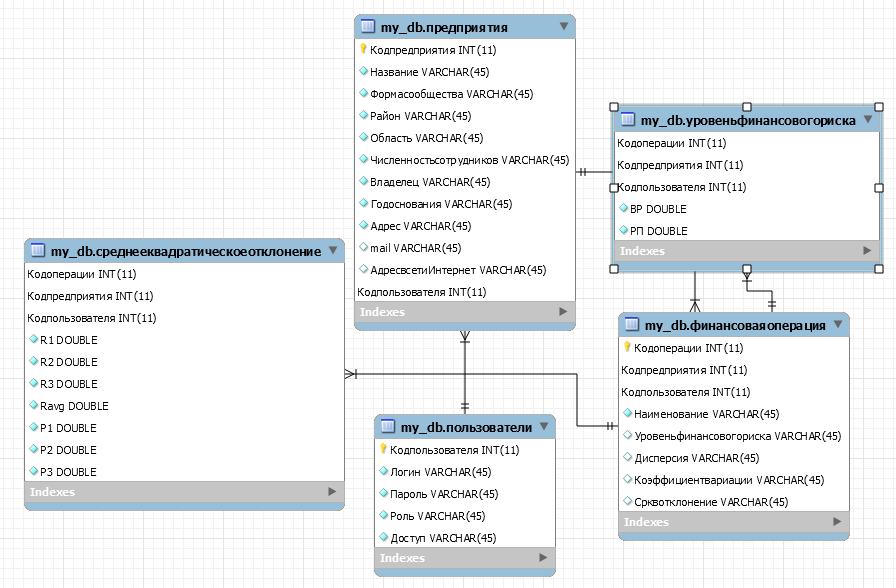


Рисунок 4.1 – Логическая модель системы оценки финансовых рисков компании

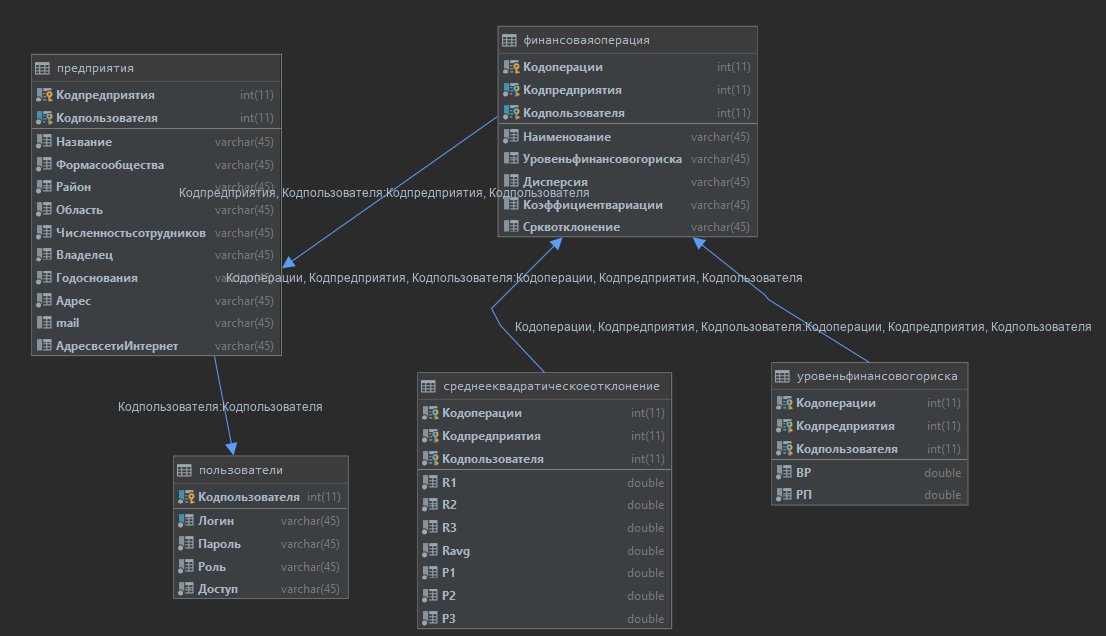
****

Рисунок 4.2 – Физическая модель системы оценки финансовых рисков компании

Сущности, на основе которых была спроектирована база данных:

* пользователи – сущность, содержащая информацию о зарегистрированном пользователе;
* предприятия – сущность, содержащая информацию о компаниях;
* финансовая операция – сущность, содержащая показатели рисков конкретной финансовой операции;
* среднеквадратическое отклонение – сущность, содержащая данные для расчета среднеквадратического отклонения, дисперсии и коэффициента вариации;
* уровень финансового риска – сущность, содержащая данные для расчета уровня финансового риска.

Данная информационная модель приведена 3 нормальной форме, а значит в ней любое поле хранит только одно значение, не ключевое поле полностью зависит от ключа и отсутствуют транзитивные зависимости.

.

1. **МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

Далее система будет рассмотрена с различных сторон и будет представлена в виде диаграмм и схем.

Были смоделированы следующие диаграммы: диаграмма вариантов использования, диаграмма состояний, диаграммы последовательностей, диаграмма развёртывания, диаграмма компонентов, диаграмма классов.

* 1. **Диаграмма вариантов использования**

В приложении актёром является финансист, который может иметь разлиные роли: пользователь, администратор и главный администратор. С повышением приоритета роли появляются новые варианты использования системы. Диаграмма вариантов использования с точки зрения главного администратора представлена в приложении А.

* 1. **Диаграмма состояний**

В данной диаграмме состояний представлен процесс авторизации пользователя в системе. Если пользователь решит не проходить авторизацию и нажмет кнопку «закрыть», будет выполнено действие по завершению работы программы, что является одним из конечных состояний данного процесса. Данная диаграмма представлена в приложении Б.

* 1. **Диаграмма классов**

Диаграмма классов – это тип статической структурной диаграммы, которая описывает структуру системы, показывая классы системы, их атрибуты, операции (или методы) и отношения между объектами.

Диаграмма классов является основным строительным блоком объектно-ориентированного моделирования. Она используется для общего концептуального моделирования систематики приложения и для подробного моделирования, переводя модели в программный код. Диаграммы классов также могут быть использованы для моделирования данных. Классы в диаграмме классов представляют как основные элементы, взаимодействия в приложении, так и классы, подлежащие программированию. Диаграмма классов данной системы представлена в приложении В.

* 1. **Диаграмма последовательностей**

Диаграмма последовательностей используется для точного определения логики сценария выполнения прецедента. Диаграммы последовательностей отображают типы объектов, взаимодействующих при исполнении прецедентов, сообщения, которые они посылают друг другу, и любые возвращаемые значения, ассоциированные с этими сообщениями.

Диаграмма последовательностей авторизации пользователя представлена в приложении Г.

* 1. **Диаграмма развертывания**

В UML диаграмма развертывания относится к структурным диаграммам, так как описывает саму систему. В данном случае диаграмма развертывания описывает физическое развертывание информации, которая генерируется программным обеспечением. Сама генерируемая информация называется артефактом.

Диаграммы состоят из нескольких трехмерных блоков, именуемых узлами. Они представляют основные программные или аппаратные компоненты. Линии от узла к узлу указывают отношения, а меньшие фигуры, содержащиеся в полях, представляют развернутые артефакты.

Диаграмма развертывания данной системы представлена в приложении Д.

1. **ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СИСТЕМЫ**
   1. **Алгоритм проверки того, заблокирован ли пользователь**

В данной системе механизм блокировки пользователя осуществляется с помощью атрибута «доступ» в сущности «пользователи». Он имеет значение «разрешен», если пользователь имеет доступ, и «закрыт», если пользователь заблокирован.

При запуске программы пользователь вводит логин и пароль. После этого, система передаёт управление серверу, который проверяет, есть ли такой пользователь в системе. Если такого пользователя нет, система выводит сообщение «Неверно введены пароль или логин.» Если же такой пользователь существует, то система проверяет атрибут «доступ». Если его значение равно «разрешен», то система передает пользователю соответствующее сообщение. Если значение атрибута равно «закрыт», то клиент получает сообщение «Эта учетная запись заблокирована администратором» и не имеет возможности войти в систему.

Блок-схема данного алгоритма представлена в приложении Е.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

При запуске программы мы видим окно авторизации, где пользователь может ввести логин и пароль, либо зарегистрироваться. При авторизации, в зависимости от роли пользователя, ему предоставляется различный функционал. В данной системе предусмотрены Пользователь, Администратор и Главный Администратор. Так как с повышением приоритета роли, расширяется функционал, в дальнейшем будем рассматривать использование программы с точки зрения Главного Администратора.

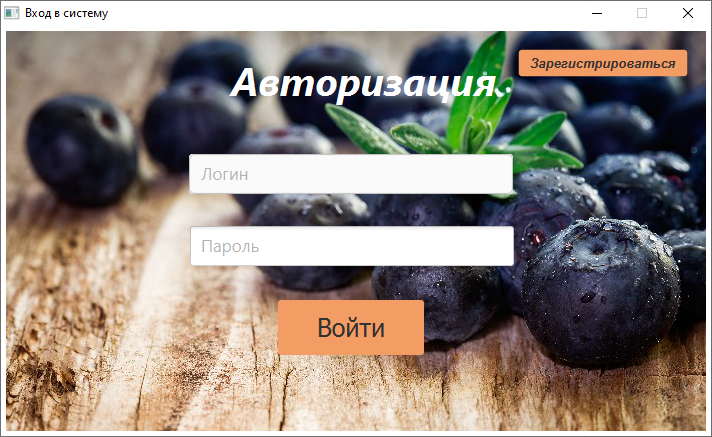


Рисунок 7.1 – Окно авторизации

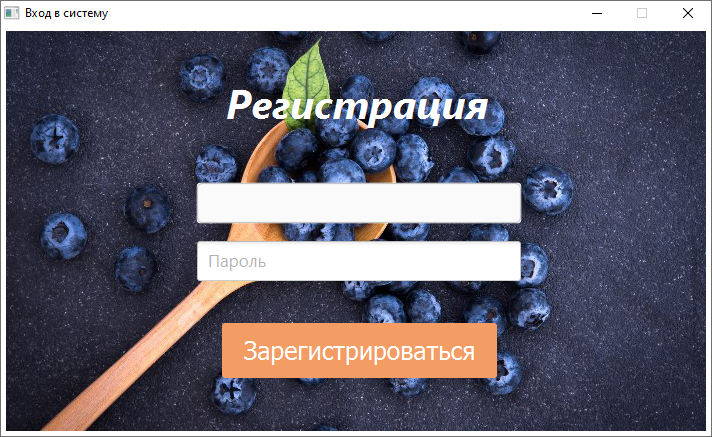


Рисунок 7.2 – Окно регистрации

После прохождения авторизации, главному администратору (менеджеру) предоставляется на выбор 2 меню: работа с пользователями и работа с предприятиями. Обычный пользователь же может работать только с предприятиями.

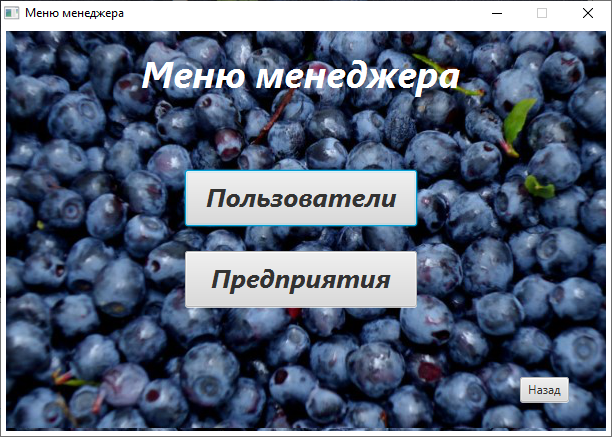


Рисунок 7.3 – Главное меню менеджера

При выборе меню работы с пользователями, менеджеру предоставляется список всех учетных записей. Он может добавить нового пользователя, выбрав пользователя в таблице, сменить его роль, заблокировать или удалить его из системы. Обычный администратор не может удалять пользователей и делать их администраторами. При блокировке пользователь лишается возможности входа в систему с выводом соответствующего сообщения.

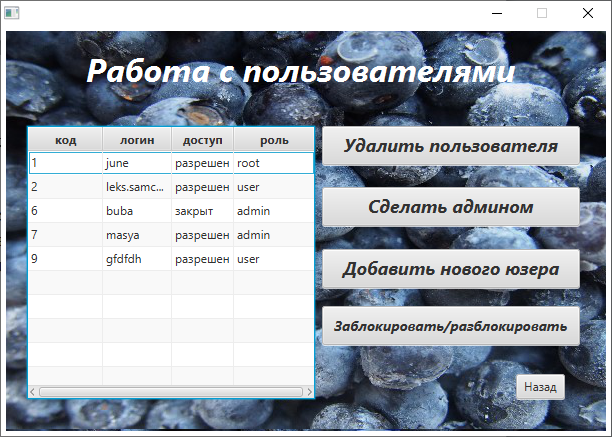


Рисунок 7.4 – Меню менеджера для работы с учетными записями

При нажатии кнопки «Добавить нового юзера» открывается новое окно, где менеджер может задать логин и пароль для нового и по нажатию кнопки «Добавить» занести его в систему.

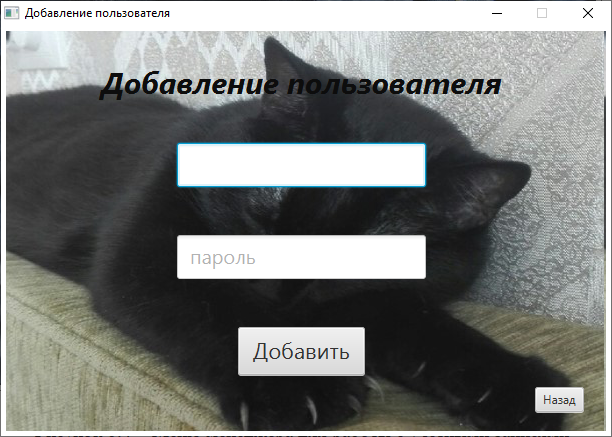


Рисунок 7.5 – Добавление нового пользователя

При выборе менеджером меню работы с предприятиями, выводится окно, в котором в виде таблицы представлена информация о всех предприятиях, содержащихся в системе. Менеджер может добавить новое предприятие, либо, выбрав одно из предприятий в таблице, удалить его или просмотреть его финансовые операции. Обычный же пользователь не может удалять предприятия.

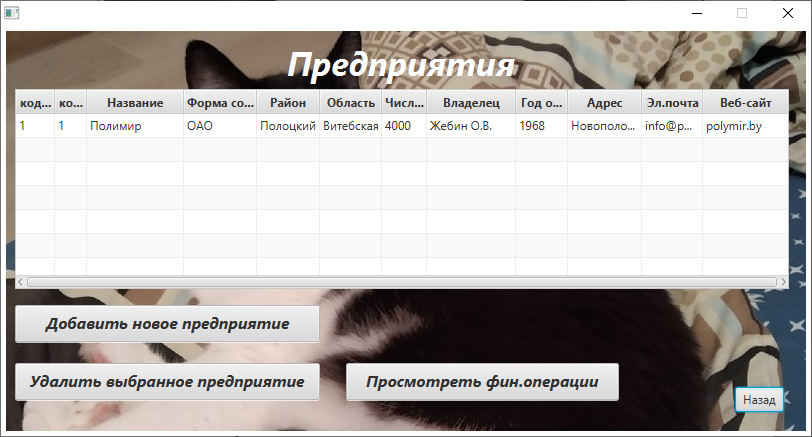


Рисунок 7.6 – Меню работы с предприятиями

При нажатии «Добавить новое предприятие» открывается окно для ввода информации о предприятии и, по нажатию кнопки «Добавить» производится занесение его в систему.

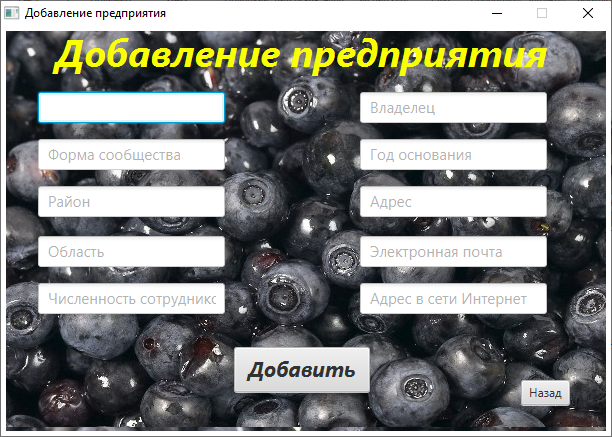


Рисунок 7.7 – Добавление нового предприятия

При нажатии кнопки «Просмотреть фин.операции» выводится окно, содержащее информацию о всех финансовых операциях выбранного предприятия.

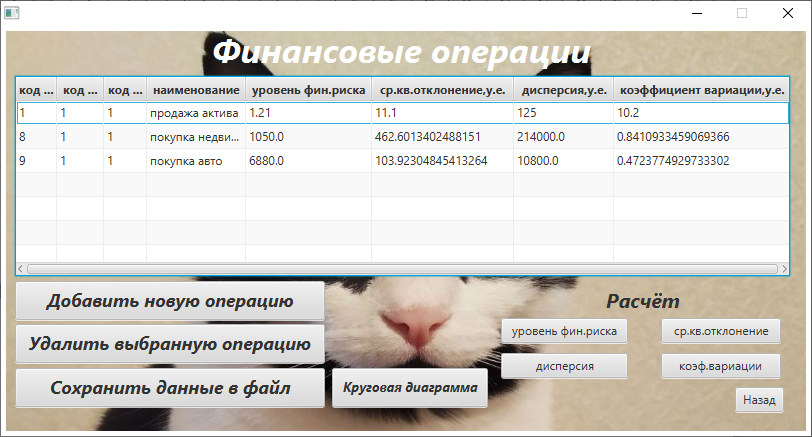


Рисунок 7.8 – Меню работы с финансовыми операциями

При нажатии «Добавить новую операцию» выводится окно, для ввода названия операции и всех параметров, необходимых для расчета финансового риска. После добавления операции, она отображается в таблице с пустыми полями «уровень фин.риска», «ср.кв.отклонение», «дисперсия» и «коэффициент вариации». При нажатии кнопки расчёта какого-либо показателя, производится расчет значения соответсвующего показателя и внесение его в таблицу. При нажатии «Сохранить данные в файл» информация о выбранной операции сохраняется в текстовый файл. При нажатии «Круговая диаграмма» выводится круговая диаграмма, отображающая соотношение показателей выбранной операции.

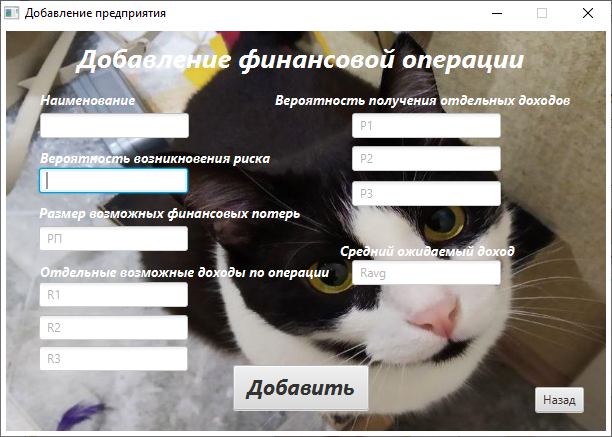


Рисунок 7.9 – Добавление новой финансовой операции

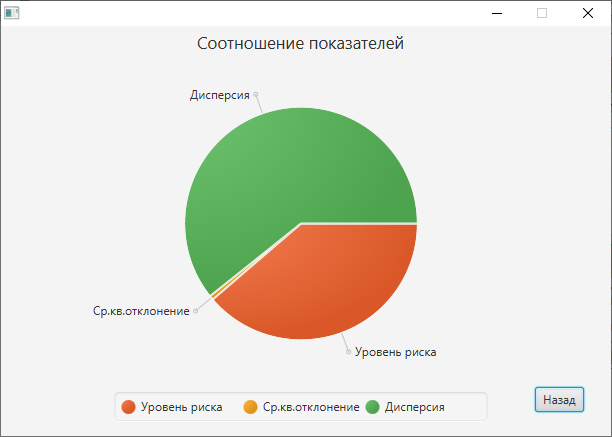


Рисунок 7.10 – Круговая диаграмма соотношения показателей финансовой операции

**ЗАКЛЮЧЕНИЕ**

В рамках данного курсового проекта было создано приложение в архитектуре клиент-сервер и организацией взаимодействия с базой данных на языке Java.

Для успешной реализации системы была разработана база данных на языке MySQL, разработана собственная иерархия классов , реализованы паттерны проектирования MVC и DAO, использованы инкапсуляция, перегрузка методов. В приложении обеспечено добавление, редактирование и удаление данных из бд, сохранение табличной информации в текстовый файл и предоставление графической информации в виде круговой диаграммы. Серверное приложение реализовано в виде консольного приложения, а клиентское – в виде оконного приложения с использованием средств создания пользовательского интерфейса(JAVAFX, SceneBuilder). Также было выполнено моделирование на основе стандарта IDEF0.

Данный программный продукт можно считать успешной реализацией курсового проекта, так как он выполняет все поставленные перед ним задачи, имеет простой, удобный, интуитивно понятный интерфейс и повышает эффективность и скорость оценки финансовых рисков предприятия.

**СПИСОК ИСПОЛЬЗУЕМЫХ РЕСУРСОВ**

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. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://discovered.com.ua/glossary/uroven-finansovogo-riska/>
5. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://searchsoftwarequality.techtarget.com/definition/use-case>
6. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://www.lucidchart.com/pages/uml-sequence-diagram>
7. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://www.lucidchart.com/pages/uml-class-diagram>
8. [Электронный ресурс]. – Электронные данные. – Режим доступа <http://forumpravo.by/files/proekt_zakona_o_personalnih_dannih.pdf>
9. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://www.lucidchart.com/pages/uml-deployment-diagram>
10. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://o7planning.org/ru/11105/javafx-piechart-tutorial>
11. [Электронный ресурс]. – Электронные данные. – Режим доступа <https://o7planning.org/ru/11079/javafx-tableview-tutorial>
12. [Электронный ресурс]. – Электронные данные. – Режим доступа https://www.studytonight.com/dbms/database-normalization.php

Приложение А

**(обязательное)**

**Диаграмма вариантов использования системы оценки финансовых рисков компании**

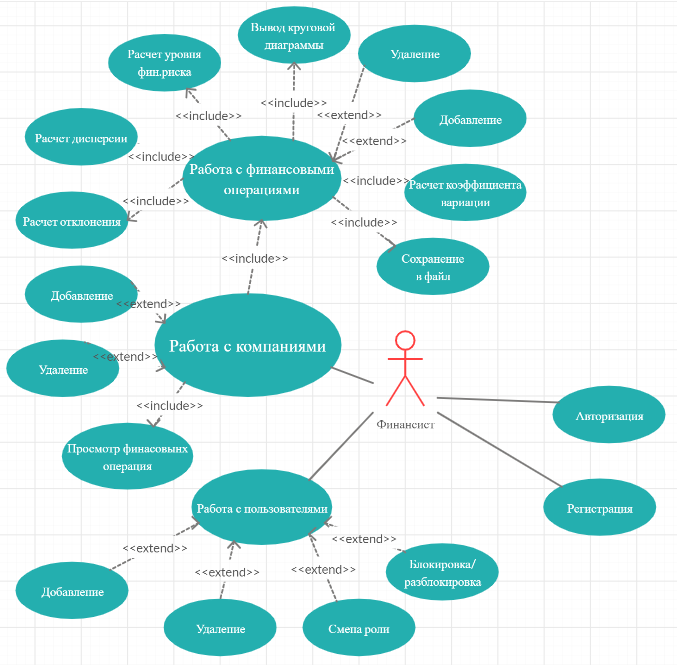


Рисунок А.1 – Диаграмма вариантов использования

Приложение Б

**(обязательное)**

**Диаграмма состояний процесса авторизации пользователя в системе оценки финансовых рисков компании**



Рисунок Б.1 – Диаграмма состояний процесса авторизации пользователя

Приложение В

**(обязательное)**

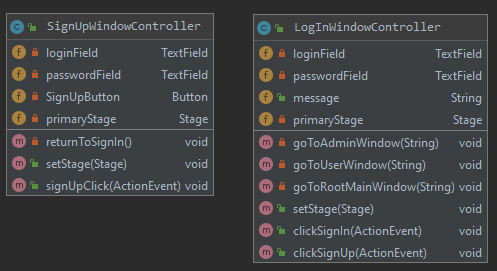


Рисунок В.1 – Диаграмма классов SignUpWindowController и LogInWindowController для регистрации и авторизации

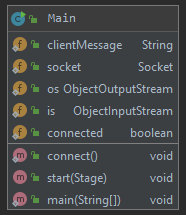


Рисунок В.2 – Диаграмма класса Main, организующего соединение клиента с сервером

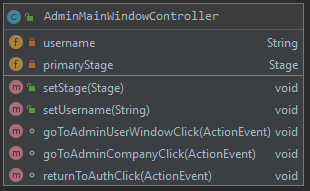


Рисунок В.3 – Диаграмма класса AdminMainWindowController, предоставляющего администратору главное меню

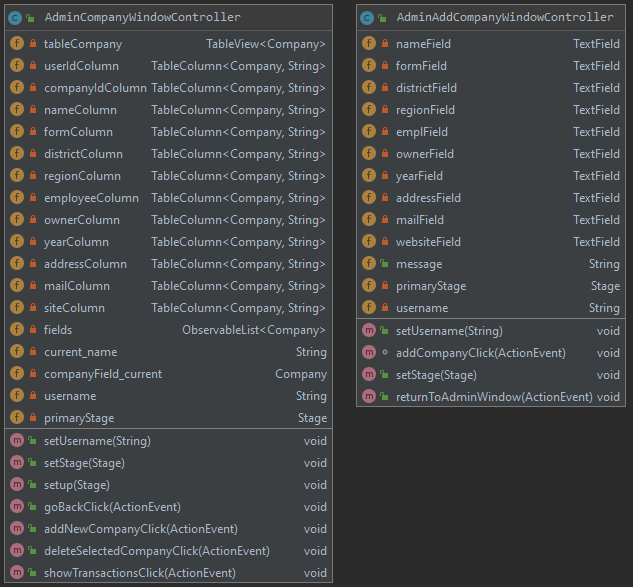


Рисунок В.4 – Диаграмма классов AdminCompanyWindowController и AdminAddCompanyWindowController, организующих работу администратора с компаниями

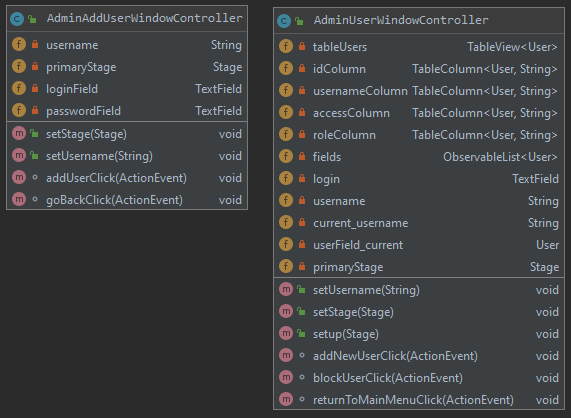


Рисунок В.5 – Диаграмма классов AdminAddUserWindowController и AdminUserWindowController, организующих работу с администратора с учетными записями

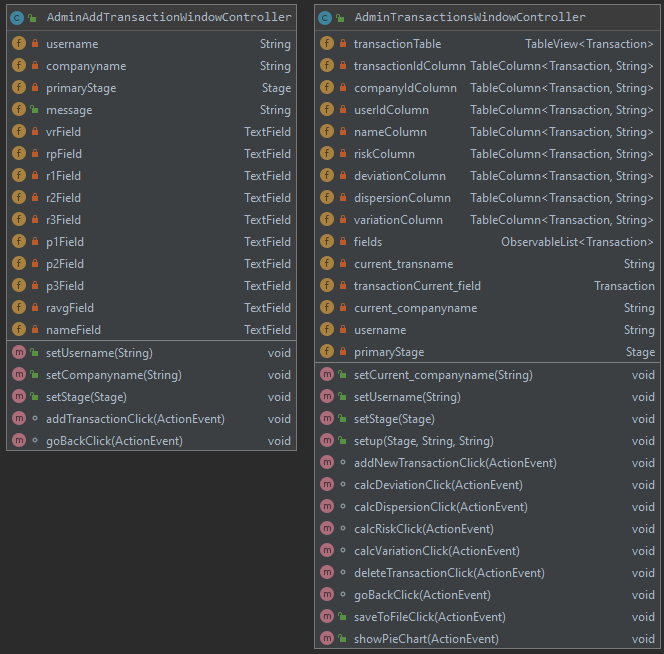


Рисунок В.6 – Диаграмма классов AdminTransactionsWindowController и AdminAddTransactionWindowController, организующих работу администратора с финансовыми операциями

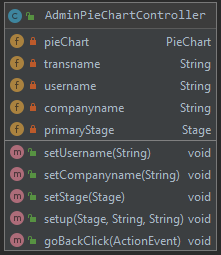


Рисунок В.7 – Диаграмма класса AdminPieChartController, предоставляющего администратору круговую диаграмму соотношения показателей финансовой операции

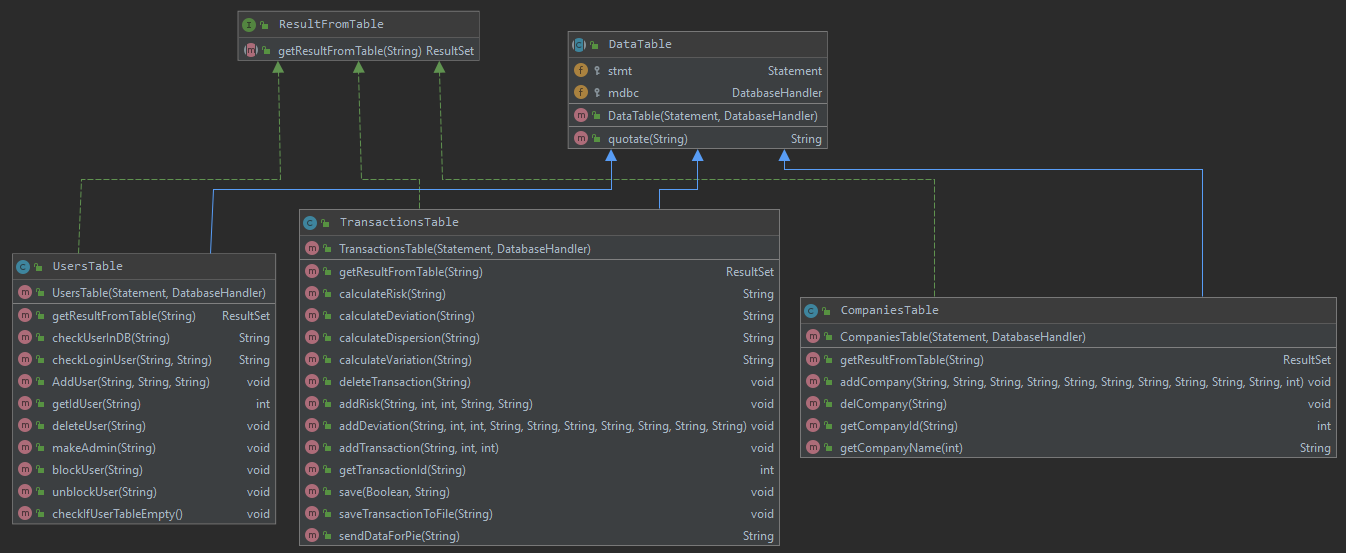


Рисунок В.8 – Диаграмма классов UsersTable, CompaniesTable и TransactionsTable, интерфейса ResultFromTable и абстрактного класса DataTable и связи между ними

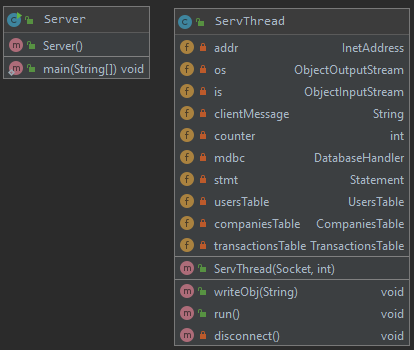


Рисунок В.9 – Диаграмма классов Server и ServThread

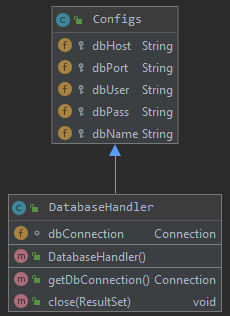


Рисунок В.10 – Диаграмма классов Configs и DatabaseHandler и связи между ними

Приложение Г

**(обязательное)**

**Диаграмма последовательностей**

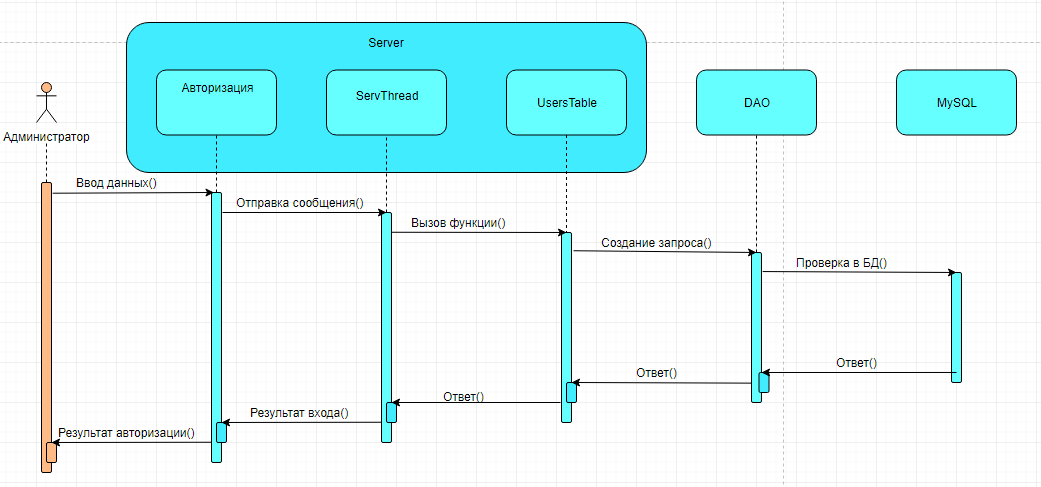


Рисунок Г.1 – Диаграмма последовательностей авторизации пользователей

Приложение Д

**(обязательное)**

**Диаграмма развертывания**

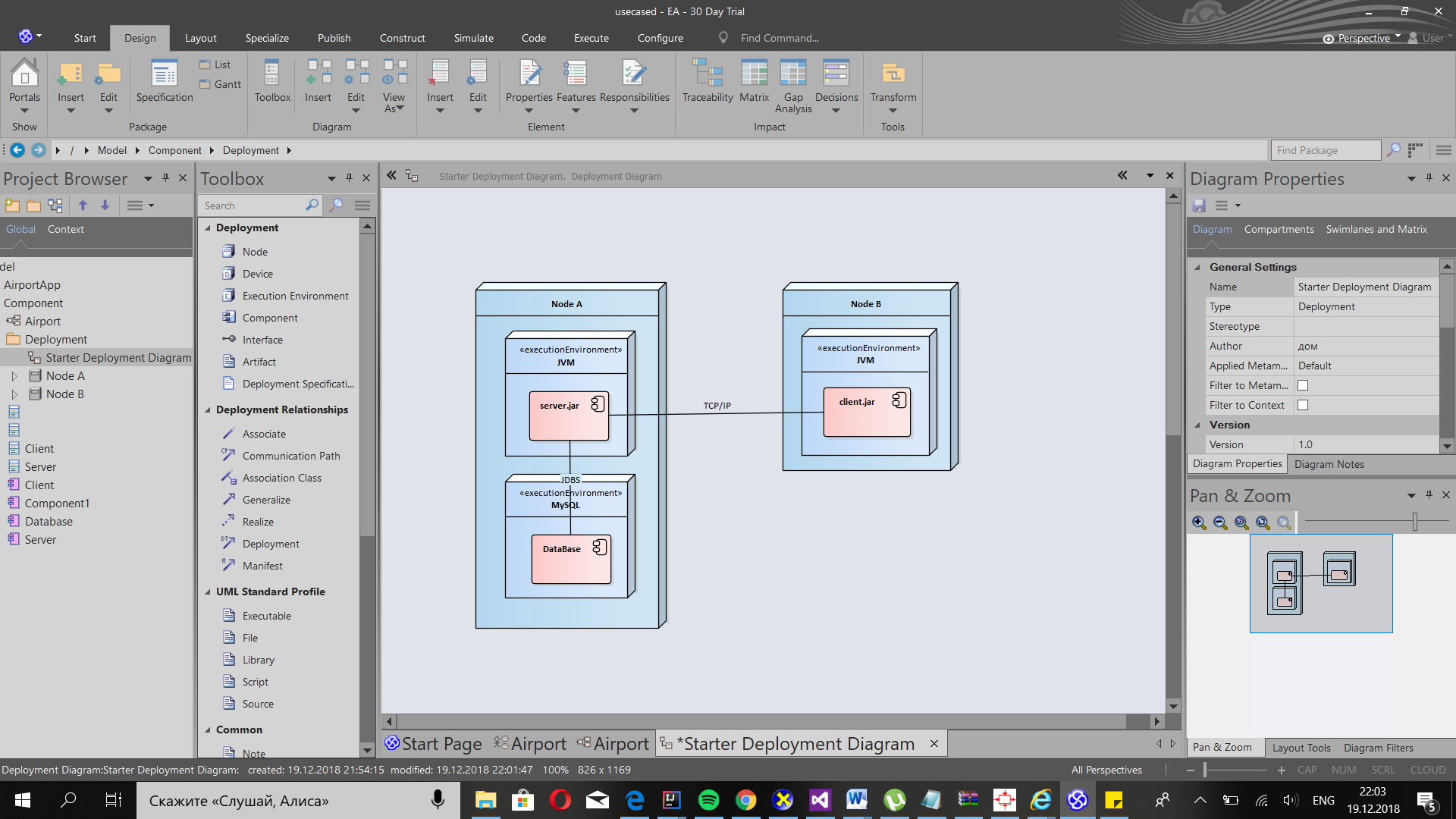


Рисунок Д.1 – Диаграмма развертывания

Приложение Е

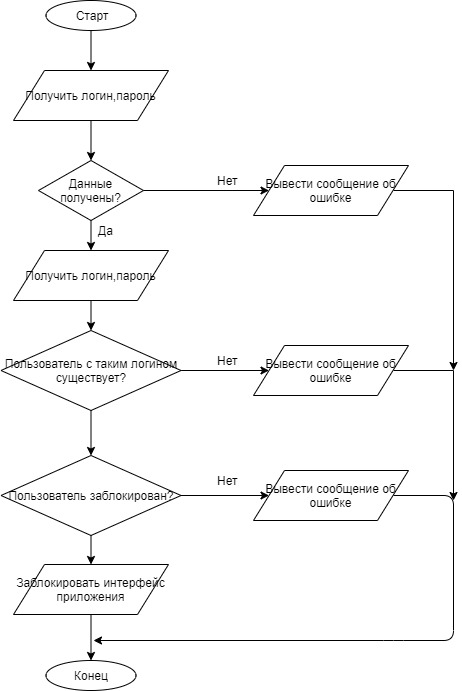


Рисунок Е.1 – Блок-схема алгоритма проверки, заблокирован ли пользователь

Приложение Ж

**(обязательное)**

**Листинг кода основных классов**

Client.src.Main

package sample;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import sample.SignUpWindow.SignUpWindowController;

import sample.logInWindow.LogInWindowController;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.InetAddress;

import java.net.Socket;

import java.net.UnknownHostException;

public class Main extends Application {

public static String clientMessage;

public static Socket socket;

public static ObjectOutputStream os;

public static ObjectInputStream is;

public static boolean connected = false;

public static void connect() {

clientMessage = "";

try {

socket = new Socket(InetAddress.getLocalHost(), 2525);

os = new ObjectOutputStream(socket.getOutputStream());

is = new ObjectInputStream(socket.getInputStream());

connected = true;

} catch (UnknownHostException var1) {

connected = false;

System.err.println("Address not available" + var1);

} catch (IOException var2) {

connected = false;

System.err.println("I/О thread error" + var2);

}

}

@Override

public void start(Stage primaryStage) throws Exception{

Main.connect();

FXMLLoader fxmlLoader=new FXMLLoader();

Parent root=null;

try {

root=fxmlLoader.load(getClass().getResource("/sample/logInWindow/LogInWindow.fxml").openStream());

} catch (IOException e) {

e.printStackTrace();

}

LogInWindowController logInWindowController=(LogInWindowController) fxmlLoader.getController();

logInWindowController.setStage(primaryStage);

primaryStage.setTitle("Вход в систему");

Scene scene = new Scene(root);

scene.getStylesheets().add("sample/logInWindow/style.css");

primaryStage.setResizable(true);

primaryStage.setScene(scene);

primaryStage.setResizable(false);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

Курсач.src.Server.ServThread

package Server;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.InetAddress;

import java.net.Socket;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class ServThread extends Thread {

private InetAddress addr;

private ObjectOutputStream os;

private ObjectInputStream is;

private String clientMessage;

private int counter;

private DatabaseHandler mdbc;

private Statement stmt;

private UsersTable usersTable;

private CompaniesTable companiesTable;

private TransactionsTable transactionsTable;

public ServThread(Socket s, int counter) throws IOException, SQLException, ClassNotFoundException {

this.counter=counter;

this.os = new ObjectOutputStream(s.getOutputStream());

this.is = new ObjectInputStream(s.getInputStream());

this.addr = s.getInetAddress();

this.mdbc = new DatabaseHandler();

Connection conn = this.mdbc.getDbConnection();

try {

this.stmt = conn.createStatement();

this.usersTable = new UsersTable(this.stmt, this.mdbc);

this.companiesTable = new CompaniesTable(this.stmt, this.mdbc);

this.transactionsTable = new TransactionsTable(this.stmt, this.mdbc);

} catch (SQLException e5) {

System.out.println(e5);

}

}

public void writeObj(String str) {

this.clientMessage = str;

try {

this.os.writeObject(this.clientMessage);

} catch (IOException e3) {

System.err.println("I/О thread error" + e3);

}

}

public void run() {

boolean i=false;

String messageToClient="";

String str;

String ThreadStop="";

try {

System.out.println("Сервер ожидает от клиента действий");

while (!ThreadStop.equals("Exit")) {

str=(String) this.is.readObject();

String[] messageParts=str.split(",");

this.usersTable.checkIfUserTableEmpty();

switch (messageParts[0]) {

case "checkLoginUser":

{

String userLogin = messageParts[1];

String userPassword = messageParts[2];

messageToClient = this.usersTable.checkLoginUser(userLogin, userPassword);

this.writeObj(messageToClient);

break;

}

case "checkUserInDB":

{

messageToClient=usersTable.checkUserInDB(messageParts[1]);

this.writeObj(messageToClient);

break;

}

case "addUser":

{

String userLogin=messageParts[1];

String userPassword=messageParts[2];

String role="user";

this.usersTable.AddUser(userLogin,userPassword,role);

break;

}

case "addCompany":

{

String name=messageParts[1];

String form=messageParts[2];

String district=messageParts[3];

String region=messageParts[4];

String employee\_number=messageParts[5];

String owner=messageParts[6];

String foundation\_year=messageParts[7];

String address=messageParts[8];

String e\_mail=messageParts[9];

String website=messageParts[10];

String userName=messageParts[11];

int userId=this.usersTable.getIdUser(userName);

this.companiesTable.addCompany(name,form,district,region,employee\_number,owner,foundation\_year,address,e\_mail,website,userId);

messageToClient="success";

this.writeObj(messageToClient);

break;

}

case "sendUserInfo":

{

ResultSet resultSet = null;

try {

resultSet=stmt.executeQuery("select \* from " + Const.USER\_TABLE);

} catch (SQLException e) {

e.printStackTrace();

}

while (resultSet.next())

{

this.os.writeObject(resultSet.getString(1));

this.os.writeObject(resultSet.getString(2));

this.os.writeObject(resultSet.getString(4));

this.os.writeObject(resultSet.getString(5));

}

this.os.writeObject("end");

break;

}

case "sendCompanyInfo":

{

ResultSet resultSet = null;

try {

resultSet=stmt.executeQuery("select \* from " + Const.COMPANY\_TABLE);

} catch (SQLException e) {

e.printStackTrace();

}

while (resultSet.next())

{

this.os.writeObject(resultSet.getString(1));

this.os.writeObject(resultSet.getString(12));

this.os.writeObject(resultSet.getString(2));

this.os.writeObject(resultSet.getString(3));

this.os.writeObject(resultSet.getString(4));

this.os.writeObject(resultSet.getString(5));

this.os.writeObject(resultSet.getString(6));

this.os.writeObject(resultSet.getString(7));

this.os.writeObject(resultSet.getString(8));

this.os.writeObject(resultSet.getString(9));

this.os.writeObject(resultSet.getString(10));

this.os.writeObject(resultSet.getString(11));

}

this.os.writeObject("end");

break;

}

case "sendTransactionsInfo":

{

int idc=this.companiesTable.getCompanyId(messageParts[1]);

ResultSet resultSet=null;

try {

resultSet=stmt.executeQuery("select \* from " + Const.TRANSACTION\_TABLE + " where (" + Const.TRANSACTION\_COMPANY\_ID + "='" + idc + "');");

} catch (SQLException e) {

e.printStackTrace();

}

while (resultSet.next())

{

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_ID));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_COMPANY\_ID));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_USER\_ID));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_NAME));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_RISK\_LEVEL));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_DEVIATION));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_DISPERSION));

this.os.writeObject(resultSet.getString(Const.TRANSACTION\_VARIATION));

}

this.os.writeObject("end");

break;

}

case "deleteUser":

{

this.usersTable.deleteUser(messageParts[1]);

break;

}

case "makeAdmin":

{

this.usersTable.makeAdmin(messageParts[1]);

break;

}

case "blockUser":

{

this.usersTable.blockUser(messageParts[1]);

break;

}

case "unblockUser":

{

this.usersTable.unblockUser(messageParts[1]);

break;

}

case "deleteCompany":

{

this.companiesTable.delCompany(messageParts[1]);

break;

}

case "deleteTransaction":

{

this.transactionsTable.deleteTransaction(messageParts[1]);

break;

}

case "addTransaction":

{

int compId=this.companiesTable.getCompanyId(messageParts[11]);

int userId=this.usersTable.getIdUser(messageParts[12]);

this.transactionsTable.addTransaction(messageParts[1],compId,userId);

this.transactionsTable.addRisk(messageParts[1],compId,userId,messageParts[2],messageParts[3]);

this.transactionsTable.addDeviation(messageParts[1],compId,userId,messageParts[4],messageParts[5],messageParts[6],messageParts[10],messageParts[7],messageParts[8],messageParts[9]);

break;

}

case "calculateRisk":

{

this.writeObj(this.transactionsTable.calculateRisk(messageParts[1]));

break;

}

case "calculateDeviation":

{

this.writeObj(this.transactionsTable.calculateDeviation(messageParts[1]));

break;

}

case "calculateDispersion":

{

this.writeObj(this.transactionsTable.calculateDispersion(messageParts[1]));

break;

}

case "calculateVariation":

{

this.writeObj(this.transactionsTable.calculateVariation(messageParts[1]));

break;

}

case "saveTransactionToFile":

{

this.transactionsTable.saveTransactionToFile(messageParts[1]);

break;

}

case "sendPieChartData":

{

this.writeObj(this.transactionsTable.sendDataForPie(messageParts[1]));

break;

}

}

}

} catch (IOException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

} finally {

disconnect();

}

}

private void disconnect() {

try {

if (this.os != null) {

this.os.close();

}

if (this.is != null) {

this.is.close();

}

System.out.println(this.addr.getHostName() + " Закрытие соединения " + this.counter + "го клиента");

} catch (IOException var5) {

var5.printStackTrace();

} finally {

this.interrupt();

}

}

}

Курсач.src.Server.TransactionsTable

package Server;

import java.io.FileOutputStream;

import java.io.IOException;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class TransactionsTable extends DataTable implements ResultFromTable {

public TransactionsTable(Statement stmt, DatabaseHandler mdbc) {

super(stmt, mdbc);

}

public ResultSet getResultFromTable(String table) {

ResultSet rs = null;

try {

rs = this.stmt.executeQuery("SELECT \* FROM " + table);

} catch (SQLException e) {

e.printStackTrace();

}

return rs;

}

public String calculateRisk(String transname)

{

double risk, possibility = 0, loss\_size = 0;

String risk\_str = null;

int idt=this.getTransactionId(transname);

try {

ResultSet resultSet = stmt.executeQuery("select " + Const.RISK\_POSSIBILITY + " from " + Const.RISK\_TABLE + " where (" + Const.RISK\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next()) {

possibility=resultSet.getDouble(Const.RISK\_POSSIBILITY);

}

resultSet=stmt.executeQuery("select " + Const.RISK\_LOSS\_SIZE + " from " + Const.RISK\_TABLE + " where (" + Const.RISK\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next()) {

loss\_size = resultSet.getDouble(Const.RISK\_LOSS\_SIZE);

}

risk=possibility\*loss\_size;

risk\_str=String.valueOf(risk);

stmt.executeUpdate(" UPDATE " + Const.TRANSACTION\_TABLE + " SET " + Const.TRANSACTION\_RISK\_LEVEL + "='" + risk\_str + "'" + " WHERE (" + Const.TRANSACTION\_ID + "='" + idt + "');");

} catch (SQLException e) {

e.printStackTrace();

}

return risk\_str;

}

public String calculateDeviation(String transname) {

double r1 = 0,r2 = 0,r3 = 0,ravg = 0,p1 = 0,p2 = 0,p3 = 0,deviation;

String deviation\_str = null;

int idt=this.getTransactionId(transname);

try {

ResultSet resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r1=resultSet.getDouble(Const.DEVIATION\_R1);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r2=resultSet.getDouble(Const.DEVIATION\_R2);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r3=resultSet.getDouble(Const.DEVIATION\_R3);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R\_AVERAGE + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

ravg=resultSet.getDouble(Const.DEVIATION\_R\_AVERAGE);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p1=resultSet.getDouble(Const.DEVIATION\_P1);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p2=resultSet.getDouble(Const.DEVIATION\_P2);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p3=resultSet.getDouble(Const.DEVIATION\_P3);

deviation=Math.sqrt(((r1-ravg)\*(r1-ravg)\*p1)+((r2-ravg)\*(r2-ravg)\*p2)+((r3-ravg)\*(r3-ravg)\*p3));

deviation\_str=String.valueOf(deviation);

stmt.executeUpdate(" UPDATE " + Const.TRANSACTION\_TABLE + " SET " + Const.TRANSACTION\_DEVIATION + "='" + deviation\_str + "' WHERE (" + Const.TRANSACTION\_ID + "='" + idt + "');"); } catch (SQLException e) {

e.printStackTrace();

}

return deviation\_str;

}

public String calculateDispersion(String transname) {

double r1 = 0,r2 = 0,r3 = 0,ravg = 0,p1 = 0,p2 = 0,p3 = 0,dispersion;

String dispersion\_str = null;

int idt=this.getTransactionId(transname);

try {

ResultSet resultSet = stmt.executeQuery("select " + Const.DEVIATION\_R1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r1 = resultSet.getDouble(Const.DEVIATION\_R1);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_R2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r2 = resultSet.getDouble(Const.DEVIATION\_R2);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_R3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r3 = resultSet.getDouble(Const.DEVIATION\_R3);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_R\_AVERAGE + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

ravg = resultSet.getDouble(Const.DEVIATION\_R\_AVERAGE);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_P1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p1 = resultSet.getDouble(Const.DEVIATION\_P1);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_P2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p2 = resultSet.getDouble(Const.DEVIATION\_P2);

resultSet = stmt.executeQuery("select " + Const.DEVIATION\_P3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p3 = resultSet.getDouble(Const.DEVIATION\_P3);

dispersion = ((r1 - ravg) \* (r1 - ravg) \* p1) + ((r2 - ravg) \* (r2 - ravg) \* p2) + ((r3 - ravg) \* (r3 - ravg) \* p3);

dispersion\_str = String.valueOf(dispersion);

stmt.executeUpdate(" UPDATE " + Const.TRANSACTION\_TABLE + " SET " + Const.TRANSACTION\_DISPERSION + "='" + dispersion\_str + "' WHERE (" + Const.TRANSACTION\_ID + "='" + idt + "');");

} catch (SQLException e) {

e.printStackTrace();

}

return dispersion\_str;

}

public String calculateVariation(String transname) {

double r1 = 0,r2 = 0,r3 = 0,ravg = 0,p1 = 0,p2 = 0,p3 = 0,deviation,coef;

String coef\_str = null;

int idt=this.getTransactionId(transname);

try {

ResultSet resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r1=resultSet.getDouble(Const.DEVIATION\_R1);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r2=resultSet.getDouble(Const.DEVIATION\_R2);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

r3=resultSet.getDouble(Const.DEVIATION\_R3);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_R\_AVERAGE + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

ravg=resultSet.getDouble(Const.DEVIATION\_R\_AVERAGE);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P1 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p1=resultSet.getDouble(Const.DEVIATION\_P1);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P2 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p2=resultSet.getDouble(Const.DEVIATION\_P2);

resultSet=stmt.executeQuery("select " + Const.DEVIATION\_P3 + " from " + Const.DEVIATION\_TABLE + " where (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

while (resultSet.next())

p3=resultSet.getDouble(Const.DEVIATION\_P3);

deviation=Math.sqrt(((r1-ravg)\*(r1-ravg)\*p1)+((r2-ravg)\*(r2-ravg)\*p2)+((r3-ravg)\*(r3-ravg)\*p3));

coef=deviation/ravg;

coef\_str=String.valueOf(coef);

stmt.executeUpdate(" UPDATE " + Const.TRANSACTION\_TABLE + " SET " + Const.TRANSACTION\_VARIATION + "='" + coef\_str + "' WHERE (" + Const.TRANSACTION\_ID + "='" + idt + "');");

} catch (SQLException e) {

e.printStackTrace();

}

return coef\_str;

}

public void deleteTransaction(String name)

{

int idt=this.getTransactionId(name);

try {

stmt.executeUpdate("DELETE FROM " + Const.DEVIATION\_TABLE + " WHERE (" + Const.DEVIATION\_TRANSACTION\_ID + "='" + idt + "');");

stmt.executeUpdate("DELETE FROM " + Const.RISK\_TABLE + " WHERE (" + Const.RISK\_TRANSACTION\_ID + "='" + idt + "');");

stmt.executeUpdate("DELETE FROM " + Const.TRANSACTION\_TABLE + " WHERE (" + Const.TRANSACTION\_ID + "='" + idt + "');");

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addRisk(String transname,int compId,int userId,String vr,String rp) {

int transId=getTransactionId(transname);

double vR,rP;

vR=Double.parseDouble(vr);

rP=Double.parseDouble(rp);

try {

this.stmt.executeUpdate("insert into " + Const.RISK\_TABLE + "(" + Const.RISK\_TRANSACTION\_ID + ","

+ Const.RISK\_COMPANY\_ID + "," + Const.RISK\_USER\_ID + "," + Const.RISK\_POSSIBILITY + ","

+ Const.RISK\_LOSS\_SIZE + ") values (" + transId + "," + compId + "," + userId + "," + vR + "," + rP + ");");

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addDeviation(String transname,int compId,int userId,String r1,String r2,String r3,String ravg,String p1,String p2,String p3) {

int transId=getTransactionId(transname);

double rd1,rd2,rd3,rdavg,pd1,pd2,pd3;

rd1=Double.parseDouble(r1);

rd2=Double.parseDouble(r2);

rd3=Double.parseDouble(r3);

rdavg=Double.parseDouble(ravg);

pd1=Double.parseDouble(p1);

pd2=Double.parseDouble(p2);

pd3=Double.parseDouble(p3);

try {

this.stmt.executeUpdate("insert into " + Const.DEVIATION\_TABLE + "(" + Const.DEVIATION\_TRANSACTION\_ID + ","

+ Const.DEVIATION\_COMPANY\_ID + "," + Const.DEVIATION\_USER\_ID + "," + Const.DEVIATION\_R1 + ","

+ Const.DEVIATION\_R2 + "," + Const.DEVIATION\_R3 + "," + Const.DEVIATION\_R\_AVERAGE + ","

+ Const.DEVIATION\_P1 + "," + Const.DEVIATION\_P2 + "," + Const.DEVIATION\_P3 + ") values (" + transId + "," + compId + "," +

userId + "," + rd1 + "," + rd2 + "," + rd3 + "," + rdavg + "," + pd1 + "," + pd2 + "," + pd3 + ");");

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addTransaction(String transname,int compId,int userId) {

try {

this.stmt.executeUpdate("insert into " + Const.TRANSACTION\_TABLE + "(" + Const.TRANSACTION\_COMPANY\_ID + ","

+ Const.TRANSACTION\_USER\_ID + "," + Const.TRANSACTION\_NAME + ")" + " values (" + compId + "," + userId + "," + this.quotate(transname) + ");");

} catch (SQLException e) {

e.printStackTrace();

}

}

public int getTransactionId(String name){

ResultSet resultSet;

int id = 0;

try {

resultSet = stmt.executeQuery("SELECT " + Const.TRANSACTION\_ID + " FROM " + Const.TRANSACTION\_TABLE + " WHERE " + Const.TRANSACTION\_NAME + "='" + name + "';");

while (resultSet.next())

id = resultSet.getInt(Const.TRANSACTION\_ID);

} catch (SQLException e) {

e.printStackTrace();

}return id;

}

public void save (Boolean bool, String text)

{

try(FileOutputStream fos=new FileOutputStream("D://Учебные материалы//Client//transaction.txt", bool))

{

byte[] buffer = text.getBytes();

fos.write(buffer, 0, buffer.length);

}

catch(IOException ex){

System.out.println(ex.getMessage());

}

}

public void saveTransactionToFile(String transname) {

int idt=getTransactionId(transname);

String companyName = null,risk = null,deviation = null,dispersion = null,variation = null,name = null;

int idc = 0;

try {

ResultSet resultSet = stmt.executeQuery("select \* from " + Const.TRANSACTION\_TABLE + " where " + Const.TRANSACTION\_ID + "='" + idt + "';");

while (resultSet.next()) {

idc = resultSet.getInt(Const.TRANSACTION\_COMPANY\_ID);

name=resultSet.getString(Const.TRANSACTION\_NAME);

risk=resultSet.getString(Const.TRANSACTION\_RISK\_LEVEL);

deviation=resultSet.getString(Const.TRANSACTION\_DEVIATION);

dispersion=resultSet.getString(Const.TRANSACTION\_DISPERSION);

variation=resultSet.getString(Const.TRANSACTION\_VARIATION);

}

resultSet=stmt.executeQuery("select " + Const.COMPANY\_NAME + " from " + Const.COMPANY\_TABLE + " where " + Const.COMPANY\_ID + "='" + idc + "';");

while (resultSet.next())

companyName=resultSet.getString(Const.COMPANY\_NAME);

resultSet=stmt.executeQuery("select " + Const.TRANSACTION\_COMPANY\_ID + " from " + Const.TRANSACTION\_TABLE + " where " + Const.TRANSACTION\_ID + "='" + idt + "';");

} catch (SQLException e) {

e.printStackTrace();

}

String text="Название предприятия: " + companyName + "\r\n";

save(false,text);

String text1="Название финансовой операции: " + name + "\r\n";

save(true,text1);

String text2="Уровень финансового риска: " + risk + "\r\n";

save(true,text2);

String text3="Среднее квадратическое отклонение: " + deviation + "\r\n";

save(true,text3);

String text4="Дисперсия: " + dispersion + "\r\n";

save(true,text4);

String text5="Коэффициент вариации: " + variation + "\r\n";

save(true,text5);

}

public String sendDataForPie(String transname) {

String data="";

String risk = null;

String dev = null;

String dis = null;

int idt=getTransactionId(transname);

try {

ResultSet resultSet = stmt.executeQuery("select " + Const.TRANSACTION\_RISK\_LEVEL + " from " + Const.TRANSACTION\_TABLE + " where " + Const.TRANSACTION\_ID + "='" + idt + "';");

while (resultSet.next())

risk=resultSet.getString(Const.TRANSACTION\_RISK\_LEVEL);

resultSet=stmt.executeQuery("select " + Const.TRANSACTION\_DEVIATION + " from " + Const.TRANSACTION\_TABLE + " where " + Const.TRANSACTION\_ID + "='" + idt + "';");

while (resultSet.next())

dev=resultSet.getString(Const.TRANSACTION\_DEVIATION);

resultSet=stmt.executeQuery("select " + Const.TRANSACTION\_DISPERSION + " from " + Const.TRANSACTION\_TABLE + " where " + Const.TRANSACTION\_ID + "='" + idt + "';");

while (resultSet.next())

dis=resultSet.getString(Const.TRANSACTION\_DISPERSION);

} catch (SQLException e) {

e.printStackTrace();

}

data=risk + "," + dev + "," + dis;

return data;

}

}

Приложение З

**(обязательное)**

**Скрипт генерации базы данных**

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema my\_db

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema my\_db

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `my\_db` DEFAULT CHARACTER SET utf8 ;

USE `my\_db` ;

-- -----------------------------------------------------

-- Table `my\_db`.`пользователи`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `my\_db`.`пользователи` (

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

`Логин` VARCHAR(45) NOT NULL,

`Пароль` VARCHAR(45) NOT NULL,

`Роль` VARCHAR(45) NOT NULL,

`Доступ` VARCHAR(45) NOT NULL DEFAULT 'разрешен',

PRIMARY KEY (`Кодпользователя`),

UNIQUE INDEX `Код пользователя\_UNIQUE` (`Кодпользователя` ASC) VISIBLE,

UNIQUE INDEX `Логин\_UNIQUE` (`Логин` ASC) VISIBLE)

ENGINE = InnoDB

AUTO\_INCREMENT = 10

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `my\_db`.`предприятия`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `my\_db`.`предприятия` (

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

`Название` VARCHAR(45) NOT NULL,

`Формасообщества` VARCHAR(45) NOT NULL,

`Район` VARCHAR(45) NOT NULL,

`Область` VARCHAR(45) NOT NULL,

`Численностьсотрудников` VARCHAR(45) NOT NULL,

`Владелец` VARCHAR(45) NOT NULL,

`Годоснования` VARCHAR(45) NOT NULL,

`Адрес` VARCHAR(45) NOT NULL,

`mail` VARCHAR(45) NULL DEFAULT NULL,

`АдресвсетиИнтернет` VARCHAR(45) NULL DEFAULT NULL,

`Кодпользователя` INT(11) NOT NULL,

PRIMARY KEY (`Кодпредприятия`, `Кодпользователя`),

INDEX `fk\_Предприятия\_Пользователи1\_idx` (`Кодпользователя` ASC) VISIBLE,

CONSTRAINT `Код пользователя`

FOREIGN KEY (`Кодпользователя`)

REFERENCES `my\_db`.`пользователи` (`Кодпользователя`))

ENGINE = InnoDB

AUTO\_INCREMENT = 12

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `my\_db`.`финансоваяоперация`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `my\_db`.`финансоваяоперация` (

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

`Кодпредприятия` INT(11) NOT NULL,

`Кодпользователя` INT(11) NOT NULL,

`Наименование` VARCHAR(45) NOT NULL,

`Уровеньфинансовогориска` VARCHAR(45) NULL DEFAULT NULL,

`Дисперсия` VARCHAR(45) NULL DEFAULT NULL,

`Коэффициентвариации` VARCHAR(45) NULL DEFAULT NULL,

`Срквотклонение` VARCHAR(45) NULL DEFAULT NULL,

PRIMARY KEY (`Кодоперации`, `Кодпредприятия`, `Кодпользователя`),

INDEX `fk\_Финансовая операция\_Предприят\_idx` (`Кодпредприятия` ASC, `Кодпользователя` ASC) VISIBLE,

CONSTRAINT `fk\_Финансовая операция\_Предприяти1`

FOREIGN KEY (`Кодпредприятия` , `Кодпользователя`)

REFERENCES `my\_db`.`предприятия` (`Кодпредприятия` , `Кодпользователя`))

ENGINE = InnoDB

AUTO\_INCREMENT = 10

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `my\_db`.`среднееквадратическоеотклонение`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `my\_db`.`среднееквадратическоеотклонение` (

`Кодоперации` INT(11) NOT NULL,

`Кодпредприятия` INT(11) NOT NULL,

`Кодпользователя` INT(11) NOT NULL,

`R1` DOUBLE NOT NULL,

`R2` DOUBLE NOT NULL,

`R3` DOUBLE NOT NULL,

`Ravg` DOUBLE NOT NULL,

`P1` DOUBLE NOT NULL,

`P2` DOUBLE NOT NULL,

`P3` DOUBLE NOT NULL,

PRIMARY KEY (`Кодоперации`, `Кодпредприятия`, `Кодпользователя`),

CONSTRAINT `fk\_Среднее квадратическое отклоне1`

FOREIGN KEY (`Кодоперации` , `Кодпредприятия` , `Кодпользователя`)

REFERENCES `my\_db`.`финансоваяоперация` (`Кодоперации` , `Кодпредприятия` , `Кодпользователя`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

-- -----------------------------------------------------

-- Table `my\_db`.`уровеньфинансовогориска`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `my\_db`.`уровеньфинансовогориска` (

`Кодоперации` INT(11) NOT NULL,

`Кодпредприятия` INT(11) NOT NULL,

`Кодпользователя` INT(11) NOT NULL,

`ВР` DOUBLE NOT NULL,

`РП` DOUBLE NOT NULL,

PRIMARY KEY (`Кодоперации`, `Кодпредприятия`, `Кодпользователя`),

CONSTRAINT `fk\_Уровень финансового риска\_Фина1`

FOREIGN KEY (`Кодоперации` , `Кодпредприятия` , `Кодпользователя`)

REFERENCES `my\_db`.`финансоваяоперация` (`Кодоперации` , `Кодпредприятия` , `Кодпользователя`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;