

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

Инженерно-экономический факультет
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

Курсовая работа по теме «Система анализа инвестиционной
привлекательности организации-эмитента»

Выполнил:
студент группы 894351
Галкин Илья Викторович

Научный руководитель
Сторожев Дмитрий Алексеевич

Минск, 2021

Содержание

Введение	4
1 Описание предметной области	6
1.1 Показатели ликвидности и платежеспособности	8
1.2 Показатели финансовой устойчивости	9
1.3 Показатели оборачиваемости активов	10
1.4 Показатели рентабельности хозяйственной деятельности.....	11
1.5 Показатели результативности работы фирмы	12
2 Постановка задачи и обзор методов ее реализации.....	14
2.1 Постановка задачи	14
2.2 Обзор используемых технологий	15
3 Функциональное моделирование системы.....	19
3.1 Описание бизнес-процесса на основе контекстной диаграммы.....	19
3.2 Описание процесса расчета на основе диаграммы последовательности	22
3.3 Диаграмма компонентов системы.....	23
3.4 Диаграмма развертывания системы	24
3.5 Диаграмма состояния объектов	25
3.6 Диаграммы классов.....	26
4 Информационная модель системы	30
5 Обоснование оригинальных решений по использованию технических и программных средств, не включенных в требования .	32
6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы.....	33
3.1 Алгоритм процесса аутентификации.....	33
3.2 Алгоритм получения одного из показателей.....	34
7 Руководство пользователя	36
7.1 Руководство по разворачиванию проекта	36
7.2 Руководство по использованию проекта.....	37
8 Результаты тестирования разработанной системы.....	41
8.1 Тестирование модуля входа в систему	41
8.2 Тестирование модуля регистрации	41

8.3 Тестирование модуля подсчета показателей.....	43
Заключение.....	44
Список использованных источников	45
Приложение А.....	46
Приложение Б.....	47

Введение

Техника и наука постоянно развиваются, что делает возможным существенно упростить и ускорить многие привычные процессы. В настоящее время повсеместно внедряются автоматизированные технологии. Они используются во всех сферах промышленности и производства, позволяют упростить технологический процесс и работу предприятия в целом.

Автоматизация систем управления подразумевает собой комплекс программных и аппаратных мероприятий и средств, позволяющих сократить количество персонала и улучшить работу систем. Особенно активно такие технологии сейчас внедряются в сферу электроэнергетики и транспорта.

Автоматизированная система не является автоматической, то есть для ее реализации и нормальной работы требуется человеческое участие.

Обычно человек-оператор выполняет основные функции управления, которые не поддаются влиянию машин. Первые автоматизированные системы появились еще в 60-е годы прошлого века, но только теперь началось их активное внедрение.

Главным предназначением АСУ является повышение производительности объекта, рост эффективности его управления, а также совершенствование методов планирования процессов управления.

АСУ активно применяются в самых разных сферах жизни и современной промышленности. В частности, они используются в системах освещения, дорожного движения, в системах информации и во всех сферах промышленного хозяйства.

Основной целью применения и использования АСУ выступает повышение эффективности и использования возможностей каждого объекта. Такие системы позволяют быстро и эффективно проводить анализ работы

объекта, на основе полученных данных специалисты могут принять определенные решения и наладить производственный процесс.

Кроме того, такие автоматизированные системы существенно ускоряют выполнение сбора и обработки данных, собранных с объекта, что позволяет снизить количество решений, принимаемых человеком.

Использование АСУ повышает уровень дисциплины и уровень контроля, так как теперь осуществлять контроль над проведением работ значительно проще и удобнее.

Автоматизированные системы повышают скорость управления, снижают затраты на выполнение многих вспомогательных операций. Самым важным последствием использования АСУ является увеличение производительности, снижение затрат и потерь в процессе производства.

Внедрение таких технологий оказывает положительное влияние на состояние отечественной промышленности и экономики, а также существенно упрощает жизнь персонала.

Однако технологии требуют финансовых вложений, причем на первых этапах деньги довольно большие, ведь наличие АСУ подразумевает смену оборудования и машин. С течением времени внедрение таких технологий окупается, а их наличие дает развитие отечественному производству. [1]

Целью данной курсовой работы является создание системы, позволяющей рассчитать основные показатели инвестиционной привлекательности организации. Это позволит собственникам компаний проанализировать динамику развития своего бизнеса на основе внутренних показателей.

Для реализации полноценной системы анализа инвестиционной привлекательности организации необходимо изучить вопросы формирования инвестиционной привлекательности, основные оценочные характеристики и различные методы оценки.

1 Описание предметной области

Инвестиционная привлекательность — это не только финансово-экономический показатель, а модель количественных и качественных показателей - оценок внешней среды (политической, экономической, социальной, правовой) и внутреннего позиционирования объекта во внешней среде, качественная оценка его финансово технического потенциала, что позволяет варьировать конечный результат.

В современной экономической литературе практически нет четкости в определении сущности инвестиционной привлекательности и правильной системы ее оценки. В общем виде можно сказать, что инвестиционная привлекательность характеризуется эффективностью инвестиционных вложений.

Инвестиционная привлекательность организации — это определенная совокупность характеристик его производственной, а также коммерческой, финансовой, в некоторой степени управленческой деятельности и особенностей того или иного инвестиционного климата, по результатам которого свидетельствуют о целесообразности и необходимости осуществления инвестиций в него.

Для анализа организации-эмитента на фактор инвестиционной привлекательности, первоначально необходимо определить виды финансовых инструментов, эмитируемых организациями и представляющие интерес для широкого круга инвесторов, а также систему показателей оценки привлекательности организации-эмитента.

Корпоративные финансовые инструменты, пользующиеся спросом со стороны инвесторов, делятся на долговые и долевыe.

Основным видом интересующих инвестора долговых ценных бумаг, выпускаемых коммерческими организациями, является облигация. Облигация — это эмиссионная ценная бумага, закрепляющая

право ее владельца на получение от эмитента облигации в предусмотренный в ней срок ее номинальной стоимости или иного имущественного эквивалента. Облигация может также предусматривать право ее владельца на получение фиксированного в ней процента от номинальной стоимости облигации либо иные имущественные права. Важнейшей инвестиционной характеристикой облигации является величина процента, которая зависит, прежде всего, от срока обращения облигации. В нормальной экономической ситуации «длинные» облигации имеют более высокие процентные выплаты, чем «короткие».

К долевым финансовым инструментам относятся акции. Акция — эмиссионная ценная бумага, закрепляющая права ее владельца (акционера) на получение части прибыли акционерного общества в виде дивидендов, на участие в управлении акционерным обществом и на часть имущества, остающегося после его ликвидации.

Акционерные общества могут выпускать обыкновенные и привилегированные (преференциальные) акции.

Владельцы обыкновенных акций приобретают ряд связанных с ними прав, таких как право на получение текущего дохода в виде дивидендов, право на участие в управлении АО путем голосования на общем собрании акционеров, право на получение информации о деятельности АО и др.

Отличительной чертой привилегированных акций является то, что ставка дивиденда по ним устанавливается в виде гарантированного фиксированного процента. Дивиденды по ним выплачиваются из чистой прибыли до выплаты дивидендов по обыкновенным акциям. Владельцы преференциальных акций участвуют в общем собрании акционеров с правом голоса при решении вопросов о реорганизации и ликвидации АО, о внесении изменений и дополнений в устав, ограничивающих права акционеров-держателей привилегированных акций, по всем вопросам компетенции собрания, начиная с собрания, следующего за годовым общим собранием акционеров, на котором не было принято решение о выплате дивиденда по

привилегированным акциям или было принято решение об их неполной выплате.

На инвестиционную привлекательность компании-эмитента ценных бумаг оказывает влияние большое количество факторов. К ним можно отнести: финансовое состояние компании, система управления организацией, номенклатура выпускаемой продукции, прозрачность компании, внешняя среда (конкуренты, партнеры), инсайдерская информация, слухи и т.д. Исходя из этого, систему показателей оценки инвестиционной привлекательности хозяйствующего субъекта составляют формализованные и неформализованные показатели.

Для оценки привлекательности долгосрочных вложений в акции и облигации потенциальный инвестор изучает имущественное положение и финансовое состояние организации-эмитента. В настоящее время в мировой учетно-аналитической практике для оценки имущественного положения и финансового состояния компаний используются десятки показателей. Классифицируя эти показатели, все их обычно подразделяют на группы, описывающие: имущественное положение компании, ее ликвидность, финансовую устойчивость, деловую активность, рентабельность, положение на рынке ценных бумаг и др.

Для оценки привлекательности долгосрочных вложений в долговые ценные бумаги используются следующие показатели. [2]

1.1 Показатели ликвидности и платежеспособности

Данные показатели являются одними из основных при анализе ИП организации-эмитента долговых ценных бумаг, так как дефолт (неспособность производить своевременные процентные и основные выплаты по долговым обязательствам) вызывается кризисом ликвидности активов, когда ликвидных активов недостаточно для того, чтобы обслужить неотложные обязательства. Ликвидность может быть оценена по-разному, но

по укрупненному балансу компании возможен лишь анализ общей ликвидности как обеспеченности краткосрочных обязательств оборотными активами. Наибольшее применение в качестве показателя ликвидности получил коэффициент текущей ликвидности, который определяется как отношение текущих активов к краткосрочным обязательствам.

Формула расчета:

$$K_{\text{ТЛ}} = \frac{OA}{KO}$$

где $K_{\text{ТЛ}}$ — коэффициент текущей ликвидности

OA — оборотные активы, тыс. руб.;

KO — краткосрочные обязательства, тыс. руб. [3]

1.2 Показатели финансовой устойчивости

Соотношение собственных и заемных средств в структуре пассивов организации является ключевым фактором для анализа ИП. Оно характеризует долю собственных средств в общей величине источников финансирования. Для определения данного соотношения рассчитывается коэффициент автономии (финансовой независимости), который характеризует степень автономии или независимости от внешнего капитала, а также долю владельцев экономического субъекта в общей сумме средств, вложенных в него

Формула расчета:

$$K_{\text{авт}} = \frac{СК}{ВБ}$$

где $K_{\text{авт}}$ — коэффициент автономии (финансовой независимости)

$СК$ — собственный капитал компании-эмитента, тыс. руб.

ВБ — валюта баланса, общая величина активов организации, тыс. руб.

[3]

1.3 Показатели оборачиваемости активов

Различным видам оборотных активов присущи различные скорости оборота. На длительность оборота активов оказывают влияние такие факторы, как вид деятельности компании (промышленность, снабжение, посредническая деятельность, сельское хозяйство); отраслевая принадлежность (тяжелая или легкая промышленность); масштабы деятельности (как правило, оборачиваемость выше на мелких предприятиях, чем на более крупных); экономическая ситуация в стране (система расчетов, вынуждающая компании отвлекать средства для предоплаты, инфляция, вынуждающая создавать большие запасы товарно-материальных ценностей); эффективность управления активами (структура активов, ценовая политика организации, методика оценки товарно-материальных ценностей). При низкой оборачиваемости средств или при ее замедлении возникает потребность в дополнительном финансировании.

Из всех показателей оборачиваемости для потенциального инвестора наиболее важен показатель оборачиваемости совокупных активов (капитала), или капиталоотдача, дающий наиболее общее представление о деловой активности коммерческой организации. Оборачиваемость совокупных активов рассчитывается по следующей формуле

$$O_{ca} = \frac{N}{(CA_{нг} + CA_{кг}) / 2}$$

где O_{ca} — оборачиваемость совокупных активов (капиталоотдача), обороты;

N — выручка от продаж за вычетом косвенных налогов, тыс. руб.;

САНГ — стоимость совокупных активов компании на начало года, тыс. руб.;

САКГ — стоимость совокупных активов компании на конец года, тыс. руб. [3]

1.4 Показатели рентабельности хозяйственной деятельности

Наиболее важную роль из данной группы показателей в оценке инвестиционной привлекательности организации-эмитента ценных бумаг играет показатель рентабельности активов (капитала). Рентабельность активов комплексно характеризует эффективность деятельности экономического субъекта. При помощи данного показателя можно оценить эффективность управления, поскольку получение высокой прибыли и достаточного уровня доходности во многом зависит от правильности выбора и рациональности принимаемых управленческих решений. По значению уровня рентабельности капитала можно оценить долгосрочное благополучие компании, то есть ее способность получать ожидаемую норму прибыли на инвестиции в достаточно длительной перспективе.

При определении рентабельности активов следует учитывать тот факт, что численное значение стоимости имущества не остается неизменным за период по причине ввода в эксплуатацию новых основных средств или их выбытия. Поэтому при исчислении рентабельности активов следует определять их среднее значение. Наиболее правильным при этом является расчет средней хронологической величины инвестированного капитала, а при отсутствии или недостаточности данных можно использовать средние арифметические значения. С учетом этого показатель рентабельности хозяйственной деятельности — коэффициент рентабельности активов (капитала) — может быть определен следующим образом:

$$K_{pa} = \frac{P_{ч}}{(A_{нг} + A_{кг})/2}$$

где K_{pa} — коэффициент рентабельности активов (капитала);

$P_{ч}$ — чистая прибыль, тыс. руб.;

$A_{нг}$ — стоимость имущества компании на начало года, тыс. руб.;

$A_{кг}$ — стоимость имущества компании на конец года, тыс. руб. [3]

1.5 Показатели результативности работы фирмы

В международной практике наиболее часто для оценки эффективности финансово-хозяйственной деятельности компаний используются показатели EVA и EBITDA.

Концепция экономической добавленной стоимости EVA рассматривает компанию как некий проект с начальным капиталом, который имеет определенную стоимость. Разница между доходностью проекта (компании) и стоимостью капитала и есть экономическая добавленная стоимость. EVA является показателем, характеризующим экономическую прибыль компании: сколько компания заработает с учетом упущенной выгоды, которую она не получит из-за невозможности вложить капитал альтернативным способом. Иными словами, экономическая добавленная стоимость (EVA) — это финансовый показатель, показывающий фактическую экономическую прибыль организации. Концепция управления, основанная на определении экономической добавленной стоимости, позволяет выяснить, достаточно ли зарабатывает компания по сравнению с альтернативными вложениями.

Другим показателем результативности работы компании является операционная прибыль, или прибыль до вычета процентов и налогов, EBIT. Операционная прибыль — это наиболее общая характеристика уровня организации и эффективности технологического процесса, лежащего в

основе ее функционирования. Специфика организации технологического процесса и техника управления им определяется управленческим персоналом (т.е. агентами собственников), а потому показатель EBIT входит в систему критериев оценки профессиональной компетентности и качества работы менеджмента. EBITDA рассчитывается как сумма чистой прибыли, чистых расходов по процентам, налога на прибыль, убытков от выбытия основных средств, убытков от обесценения, расходов по увеличению обязательств по выбытию активов, износа и амортизации, за вычетом прибыли (убытков) от финансовых вложений и дохода от прекращенной деятельности.

Представленный выше набор показателей, является достаточно полным и дает целостную картину привлекательности коммерческой организации. Финансовые коэффициенты имеют одинаковую направленность (обладают положительной динамикой) и рассчитываются по данным публичной отчетности. Расчетные показатели дают возможность оценивать ИП компании как в пространстве (в сравнении с другими компаниями отрасли), так и во времени.

На основе отобранных показателей представляется возможным расчет итогового (интегрального) показателя оценки инвестиционной привлекательности компании, используя следующую формулу:

$$K_{ип} = \sqrt{\sum_{i=1}^n \left(1 - \frac{X_{ij}}{X_{iet}}\right)^2}$$

где $K_{ип}$ — интегральный показатель инвестиционной привлекательности компании-эмитента ценных бумаг;

n — количество финансовых коэффициентов, участвующих в расчете;

X_{ij} — значение финансового коэффициента i у организации j ;

X_{iet} — эталонное значение финансового коэффициента i . [3]

2 Постановка задачи и обзор методов ее реализации

2.1 Постановка задачи

По итогам работы должно быть разработано приложение, позволяющее рассчитать показатели инвестиционной привлекательности организаций-эмитентов по заданным параметрам. Приложение должно быть интуитивно понятным для пользователя и безопасным для использования.

Приложение должно быть основано на архитектуре клиент-сервер, поддерживать механизмы аутентификации и авторизации и поддерживать логирование результатов работы.

В приложении должен быть реализован механизм обработки исключительных ситуаций и некорректного ввода, с выводом соответствующих ошибок информационных сообщений.

Ролевая модель в приложении должна быть представлена двумя типами:

- 1) Пользователь
- 2) Администратор

Пользователю с ролью администратор должны быть доступны все функции обычного пользователя.

Приложение должно поддерживать всю функциональность диаграммы вариантов использования.

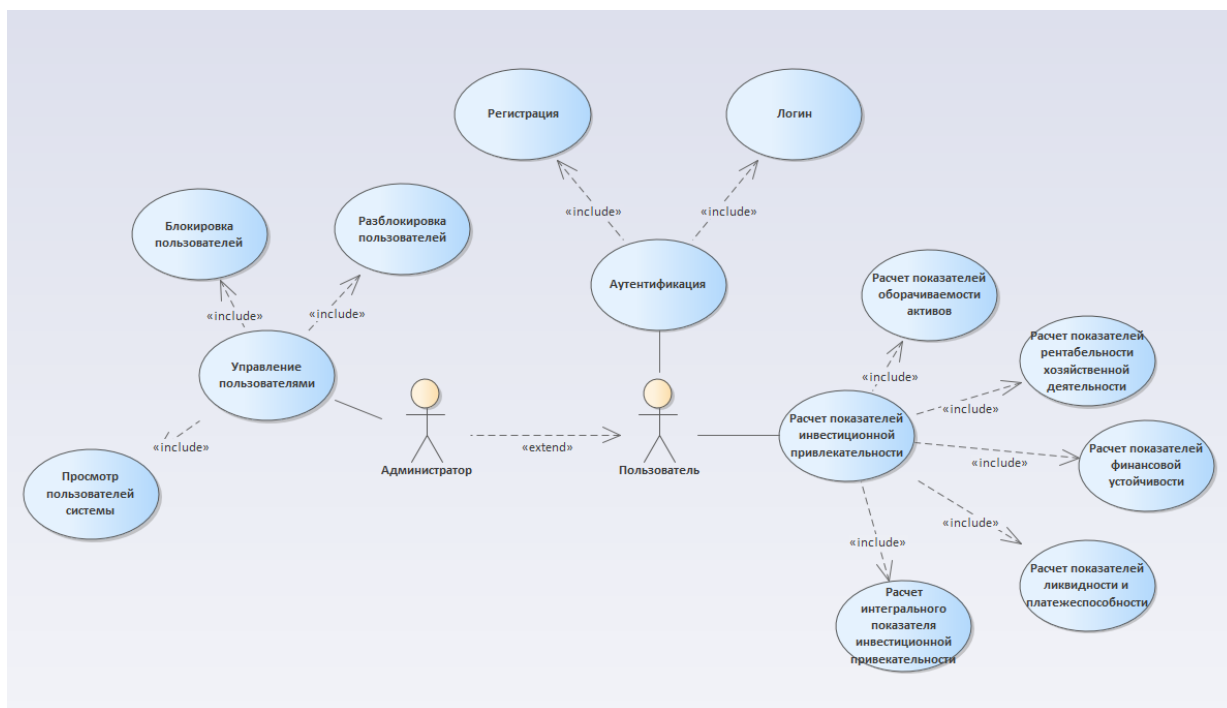


Рисунок 2.1 – Use-case диаграмма приложения

2.2 Обзор используемых технологий

Приложение было реализовано на языке программирования Java с использованием фреймворка Spring.

Язык Java имеет массу достоинств, благодаря которым он хорошо подходит под разработку подобной системы:

- 1) Простота – у языка четкие синтаксические правила и понятная семантика.
- 2) Объектно-ориентированный подход – в центре внимания находятся объекты, что позволяет легко смоделировать систему по заданным требованиям.
- 3) Безопасность – важный параметр в сетевых приложениях. Обойти или взломать механизмы защиты крайне сложно.
- 4) Производительность – язык Java использует JIT компилятор, который ускоряет работу приложения в разы.

- 5) Надежность – программы на данном языке программирования стабильно работают в любых условиях
- 6) Независимость от аппаратной части и ОС – приложения на языке Java могут быть развернуты на практически любой среде благодаря исполняющей среде и JVM.
- 7) Поддержка сетевых возможностей – язык Java поддерживает самые распространенные сетевые протоколы, такие как: FTP, HTTP, TCP/IP.

В данном проекте был использован фреймворк Spring. Основная задача Spring – упрощение разработки приложения для разработчика. Фреймворк Spring предоставляет:

- 1) Заданную архитектуру приложения. Spring предоставляет каркас приложения и диктует правила построения приложения. В состав Spring входит много подпроектов, решающих определенные задачи (SpringMVC, SpringSecurity, SpringData). Разработчик сам решает, какие из подпроектов использовать.
- 2) Реализует внедрение зависимостей. Это абстрактный принцип для написания слабо связанного кода. Суть которого в том, что каждый компонент системы должен быть как можно более изолированным от других, не полагаясь в своей работе на детали конкретной реализации других компонентов.
- 3) Поддержку IoC контейнера. Spring берет задачу инициализацию и хранение необходимых зависимостей на себя. Разработчику необходимо только контролировать регистрацию объектов в контейнер. IoC контейнер хранит все зависимости в себе и позволяет получить необходимую зависимость используя тип объекта как ключ.
- 4) Spring умеет связывать объекты без прямого участия разработчика. Это позволяет уменьшить количество кода при определении зависимостей компонентов.

5) В Spring настройка компонентов вынесены в отдельные файлы, что облегчает процесс замены реализации.

В реализованном проекте, помимо основного модуля Spring, были использованы:

- SpringData – модуль, упрощающий работу с базами данных
- SpringSecurity – модуль, упрощающий разработку процессов авторизации и аутентификации
- SpringMVC – модуль, упрощающий реализацию архитектурного паттерна MVC.

Серверная часть приложения была построена на основе архитектурного паттерна MVC (Model-View-Controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач.

- 1) Model – отвечает за данные, определяет структуру приложения.
- 2) View – отвечает за взаимодействие с пользователем. Содержит внешний вид приложения и способы его использования
- 3) Controller – отвечает за связь Model и View. Определяет действия сервера на действия пользователя

В данном приложении модель данных представлена в виде набора классов, которые следуют правилам построения компонентов JavaBeans. Представления были реализованы при помощи технологии JSP, а также с использованием HTML, CSS и JavaScript. Контроллеры были реализованы с использованием технологии Servlet.

В качестве хранилища данных была выбрана СУБД PostgreSQL. Это свободная объектно-реляционная система управления базами данных. Базируется на языке SQL и простая в освоении. Для работы с СУБД был

использован драйвер JDBC. Сама база данных была приведена к третьей нормальной форме и для нее был реализован генерирующий SQL-скрипт.

Для разворачивания приложения был использован контейнер сервлетов Tomcat.

3 Функциональное моделирование системы

3.1 Описание бизнес-процесса на основе контекстной диаграммы

Система анализа инвестиционной привлекательности организации-эмитента должна помогать пользователям рассчитать показатели инвестиционной привлекательности. Для того, чтобы достичь поставленной цели, пользователю необходимо знать годовые показатели своей организации и быть зарегистрированным в системе. Основная работа программы будет обеспечиваться при помощи базы данных и формул для подсчета, хранящихся на сервере. Взаимодействие с приложением будет вести пользователь при помощи веб-интерфейса.

Контекстная диаграмма бизнес-процесса изображена на рисунке 3.1.

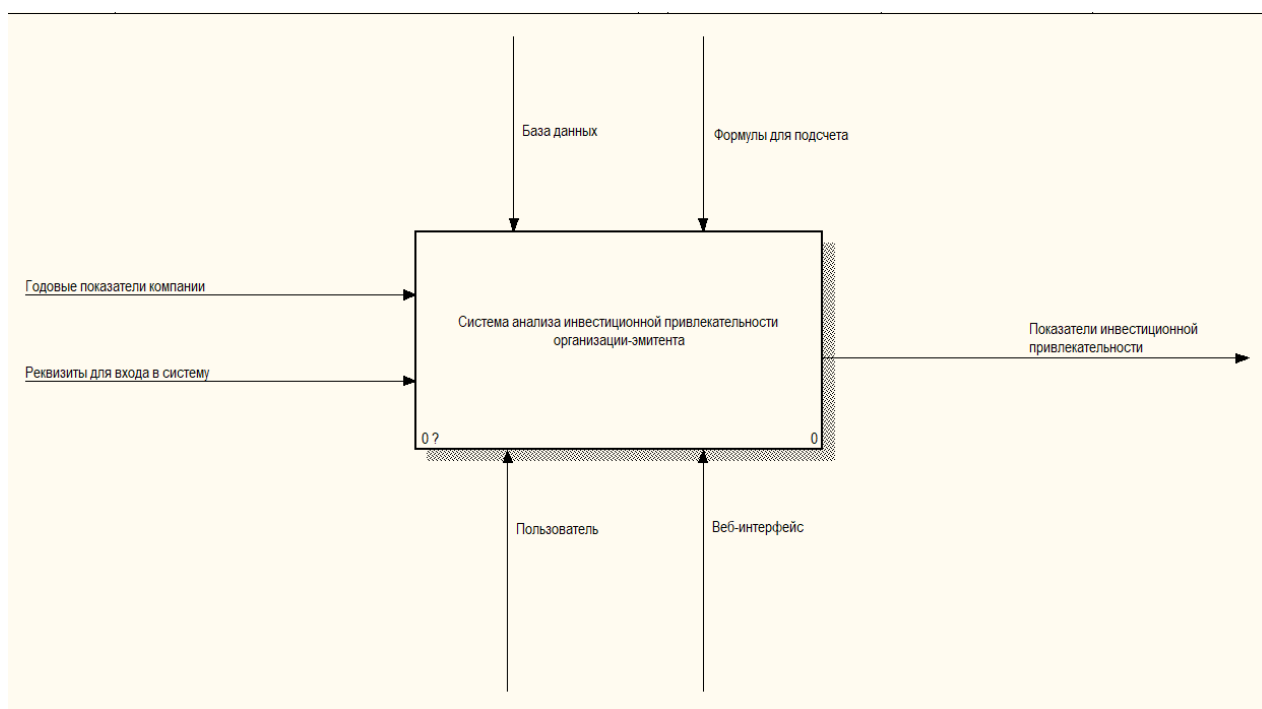


Рисунок 3.1 – Контекстная диаграмма бизнес-процесса приложения

Сам процесс анализа инвестиционной привлекательности организации-эмитента при помощи данного приложения можно декомпозировать на 3 блока:

- Аутентификация
- Расчет показателей инвестиционной привлекательности
- Сохранение и получение результатов расчета

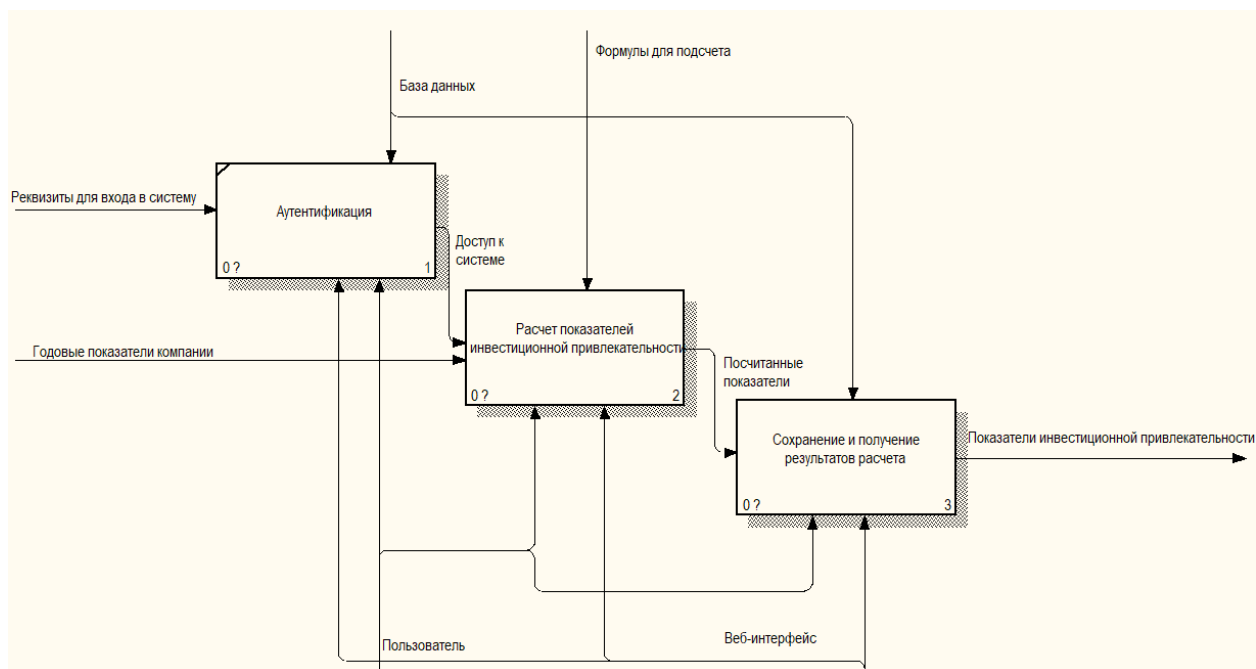


Рисунок 3.2 – Декомпозиция процесса получения показателей инвестиционной привлекательности

На этапе аутентификации пользователю необходимо ввести реквизиты для входа в систему для того, чтобы получить доступ к системе.

После того, как доступ был получен, пользователь может рассчитать один из нескольких показателей инвестиционной привлекательности.

Процесс расчета показателей инвестиционной привлекательности представлен на рисунке 3.3.

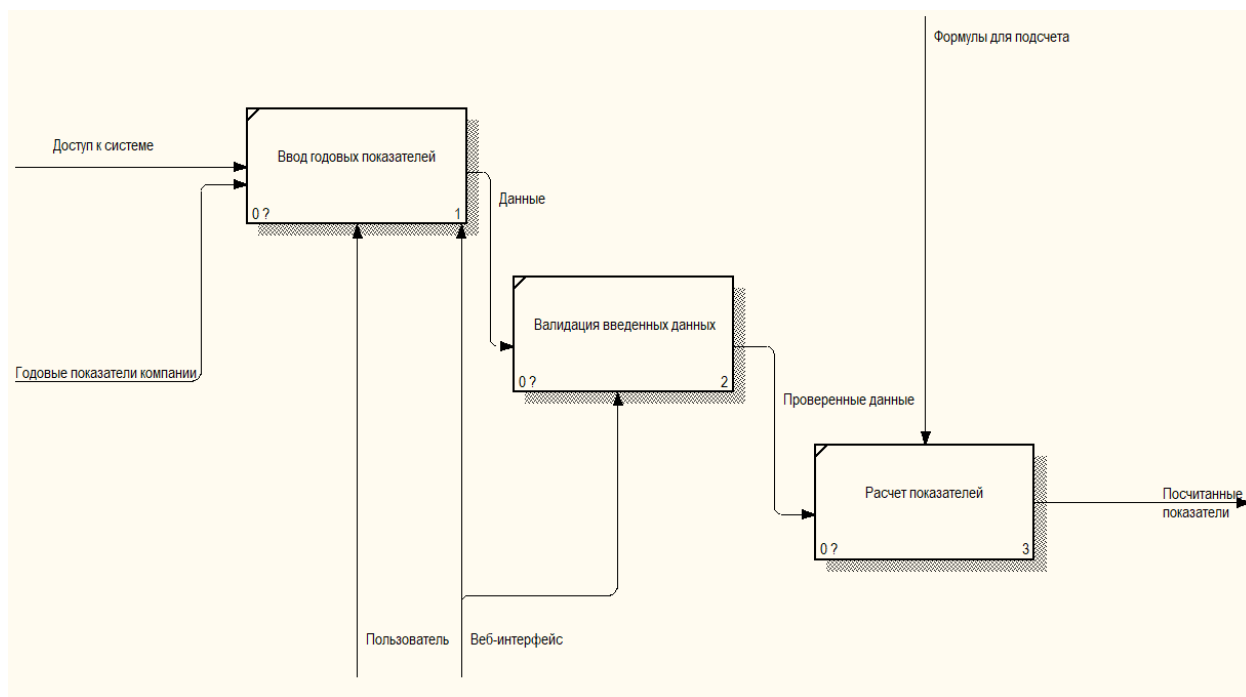


Рисунок 3.3 – Декомпозиция процесса расчета показателей инвестиционной привлекательности

Прежде всего, пользователь вводит годовые показатели своей компании. Далее, система проверяет введенные данные и, если данные корректны, система рассчитывает показатели по определенным формулам.

После того, как показатели были посчитаны, система сохраняет полученные результаты в таблицу с аудит информацией и возвращает результат пользователю в браузер. Далее, пользователь переносит полученные данные в свою локальную систему. Данный процесс проиллюстрирован на рисунке 3.4.

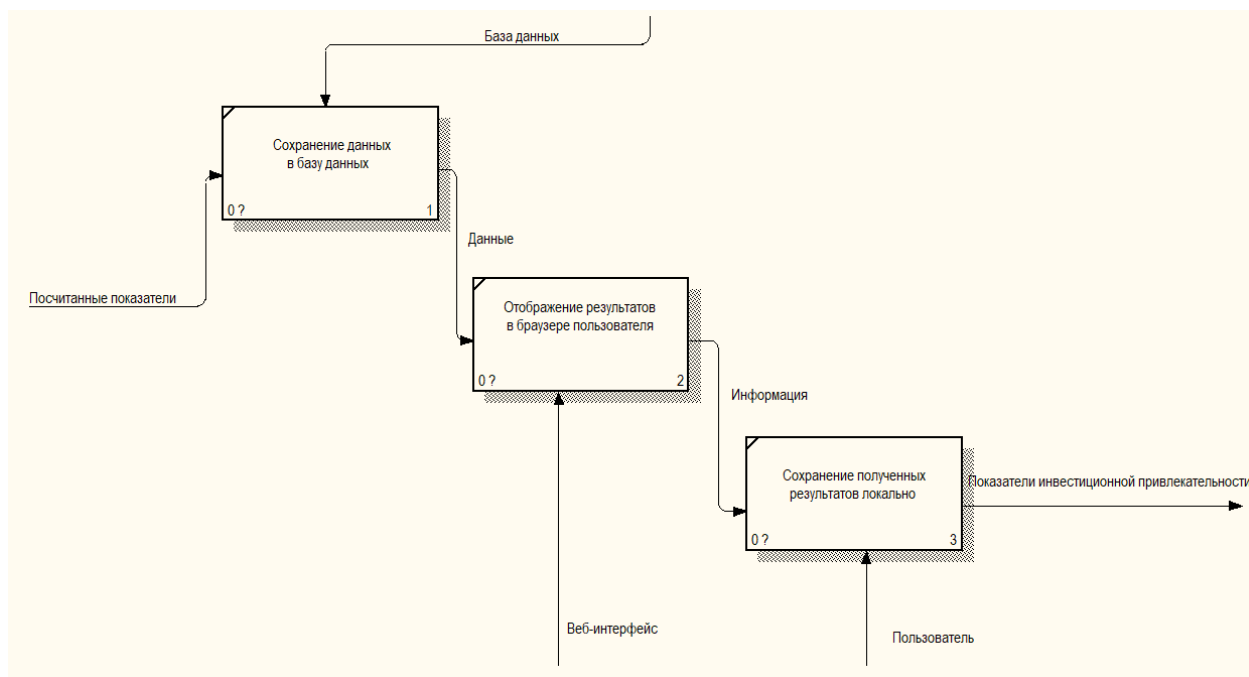


Рисунок 3.4 – Декомпозиция процесса сохранения и получения результатов расчета

3.2 Описание процесса расчета на основе диаграммы последовательности

Диаграмма последовательности отображает взаимосвязанную работу нескольких компонентов бизнес-процесса. Они представляют собой иллюстрацию потока событий в результате использования приложения.

На рисунке 3.5 изображена диаграмма последовательности расчета показателя оборачиваемости актива.

Для данного приложения можно выделить следующие компоненты:

- 1) Пользователь
- 2) Браузер
- 3) Контроллер
- 4) Сервис для подсчета показателей
- 5) База данных

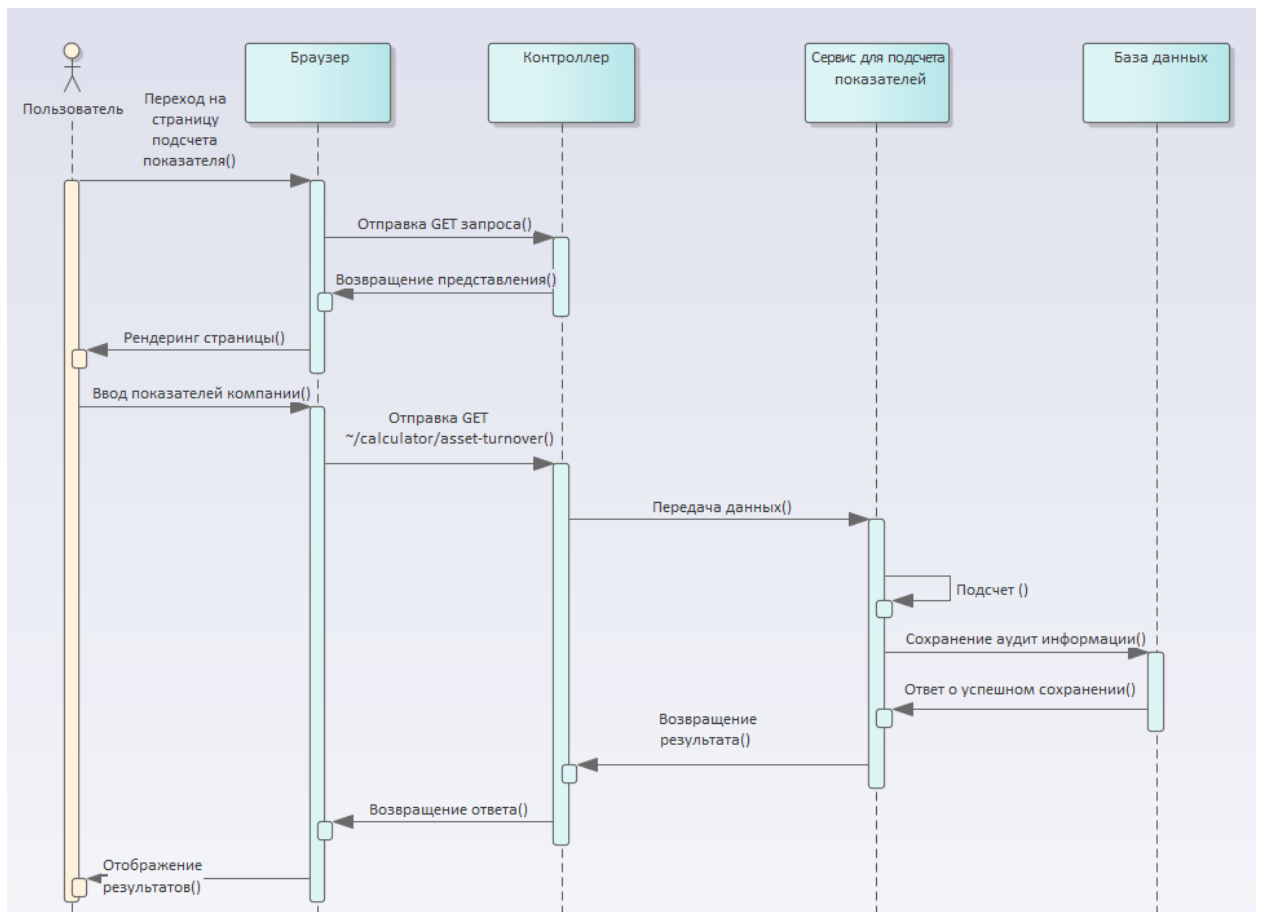


Рисунок 3.5 – Диаграмма последовательности процесса подсчета показателей

3.3 Диаграмма компонентов системы

Диаграмма описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними. [4]

На рисунке 3.6 изображена диаграмма компонентов разработанной системы.

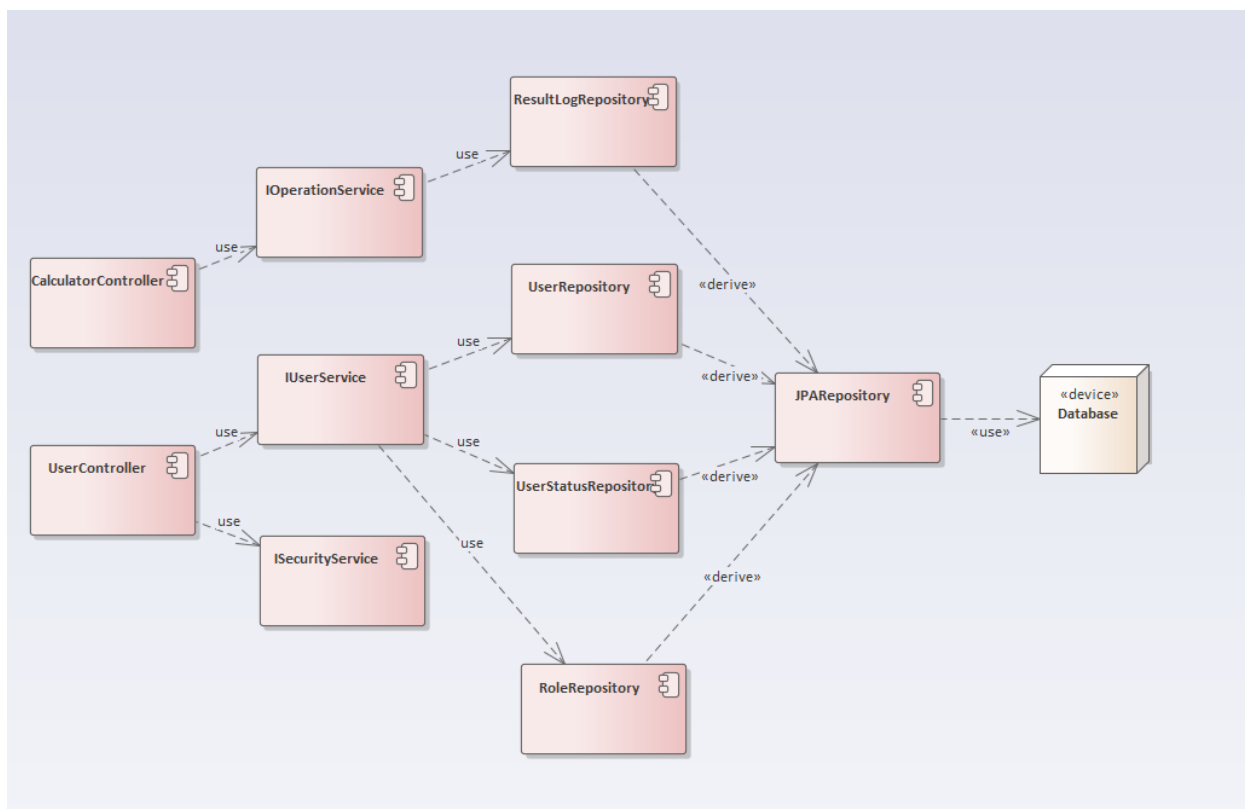


Рисунок 3.6 – Диаграмма компонентов системы

3.4 Диаграмма развертывания системы

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения (runtime). При этом представляются только компоненты-экземпляры программы, являющиеся исполнимыми файлами или динамическими библиотеками. Те компоненты, которые не используются на этапе исполнения, на диаграмме развертывания не показываются. [4]

На рисунке 3.7 изображена диаграмма развертывания разработанной системы.

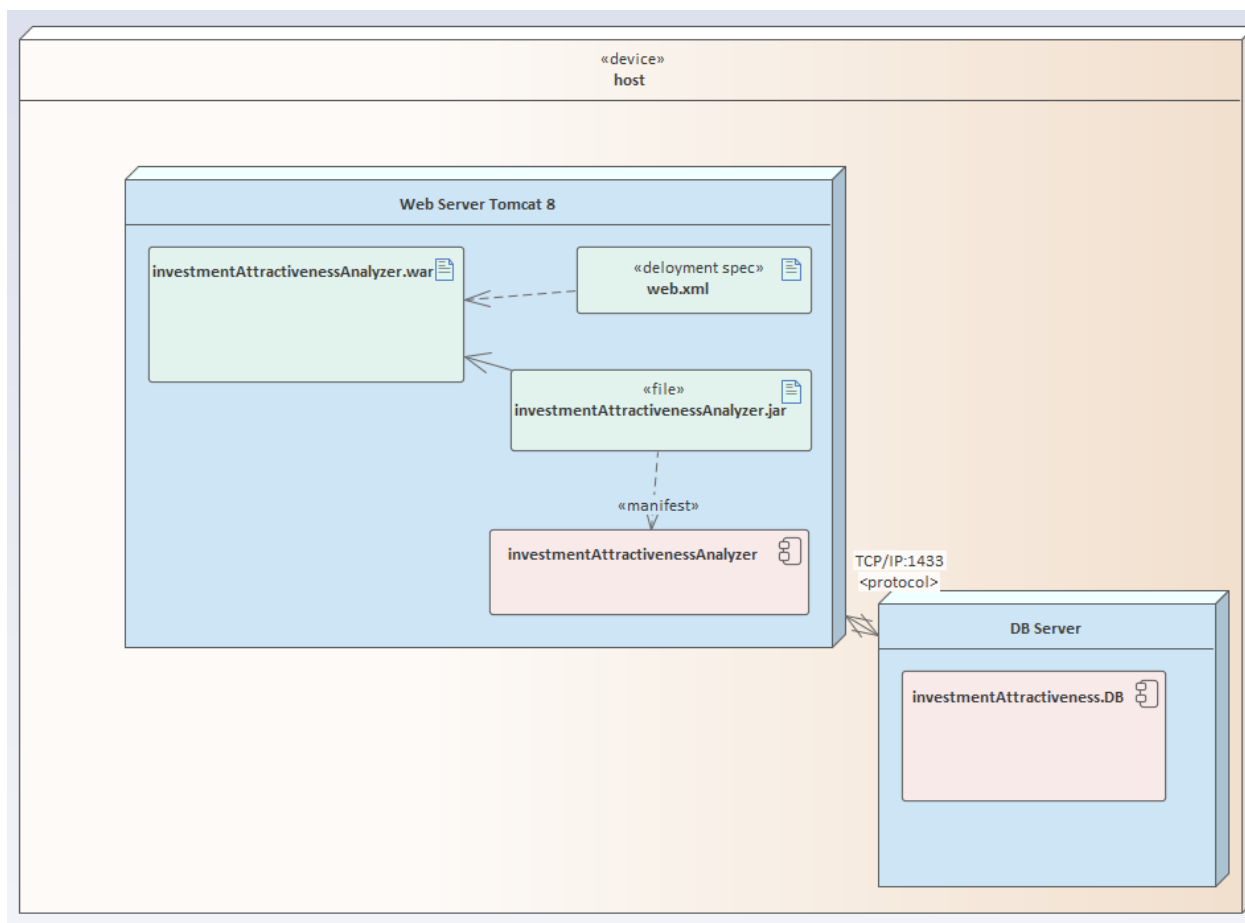


Рисунок 3.7 – Диаграмма развертывания системы

3.5 Диаграмма состояния объектов

Диаграмма состояния объектов показывает изменения состояния объектов в контексте определенного действия. В качестве изучаемой ситуации была выбрана блокировка и разблокировка пользователя администратором системы. Диаграмма состояния объектов изображена на рисунке 3.8. [4]

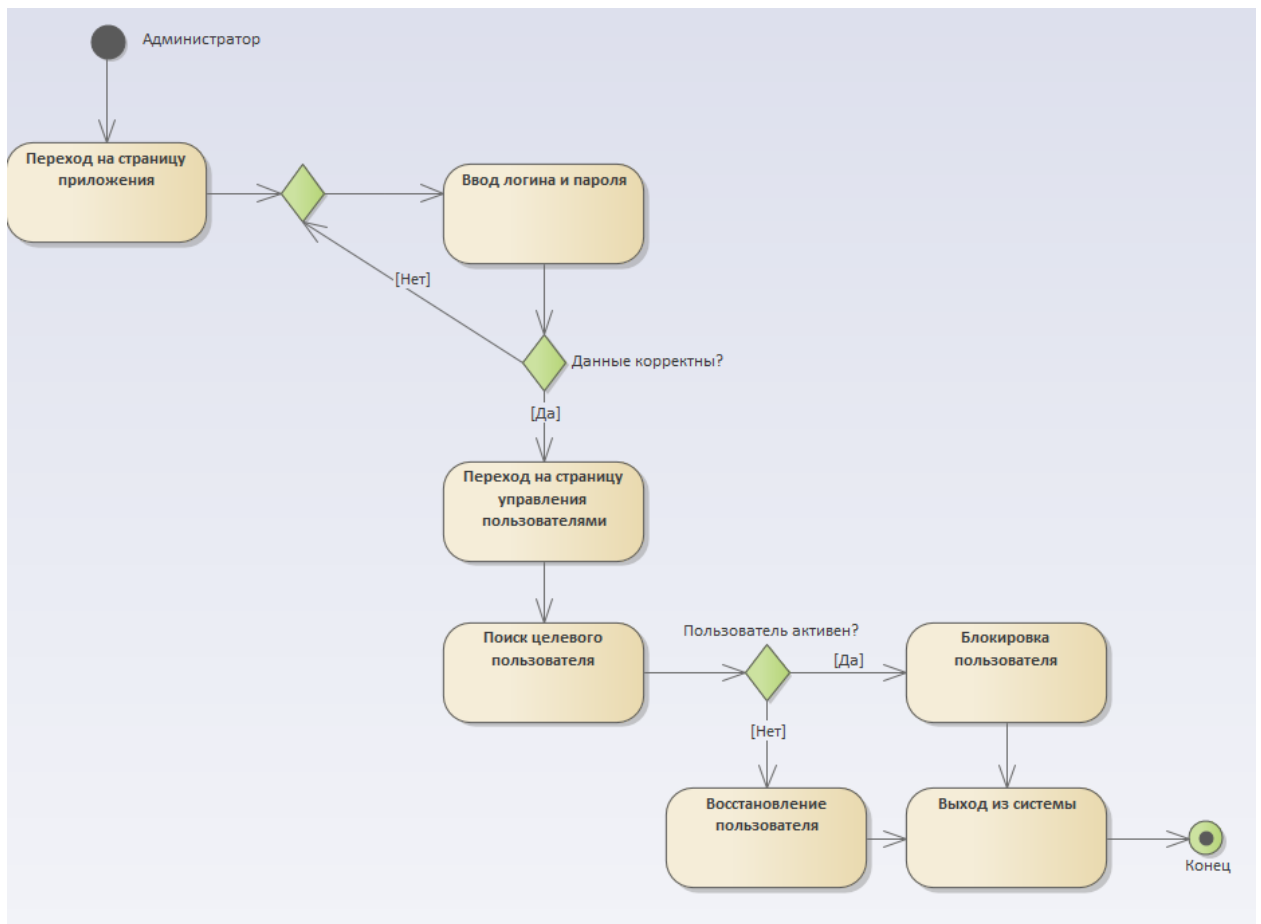


Рисунок 3.8 – Диаграмма состояния объектов

3.6 Диаграммы классов

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы. На этом этапе принципиально знание ООП подхода и паттернов проектирования. [4]

Основная бизнес-логика работы приложения завязана на взаимодействии между сервисами приложения и репозиториями, которые содержат все методы доступа к таблице связанной сущности в базе данных.

Все репозитории системы наследуются от `JpaRepository`, предоставляющий основные методы доступа к данным. В случае, если на репозиторий нужно добавить дополнительную логику, при помощи специального синтаксиса в названии метода, можно добавлять различные реализации взаимодействия в БД.

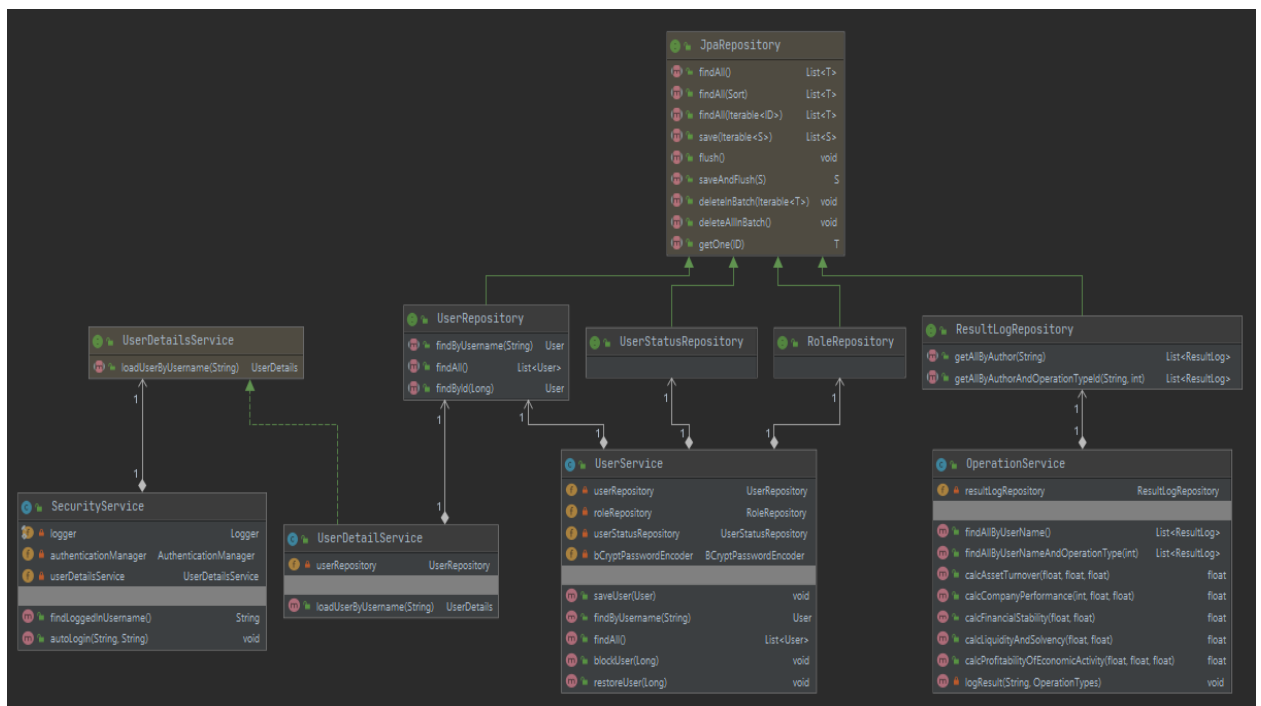


Рисунок 3.9 – Диаграмма классов репозитория и связанных с ними сервисами

На рисунке 3.10 представлена диаграмма классов-моделей. Модели представляют собой отражение данных, хранящихся в базе данных. При помощи специальных аннотаций `Hibernate` каждое поле модели соотносится с колонкой на таблице в БД, а каждая модель – с таблицей.

В данном приложении основными моделями являются `User` и `ResultLog`.

Модель User отражает абстрактного пользователя системы, а ResultLog – аудит данные системы. Модели UserStatus и Role являются моделями-справочниками, хранящими подробную информацию о связанной сущности.

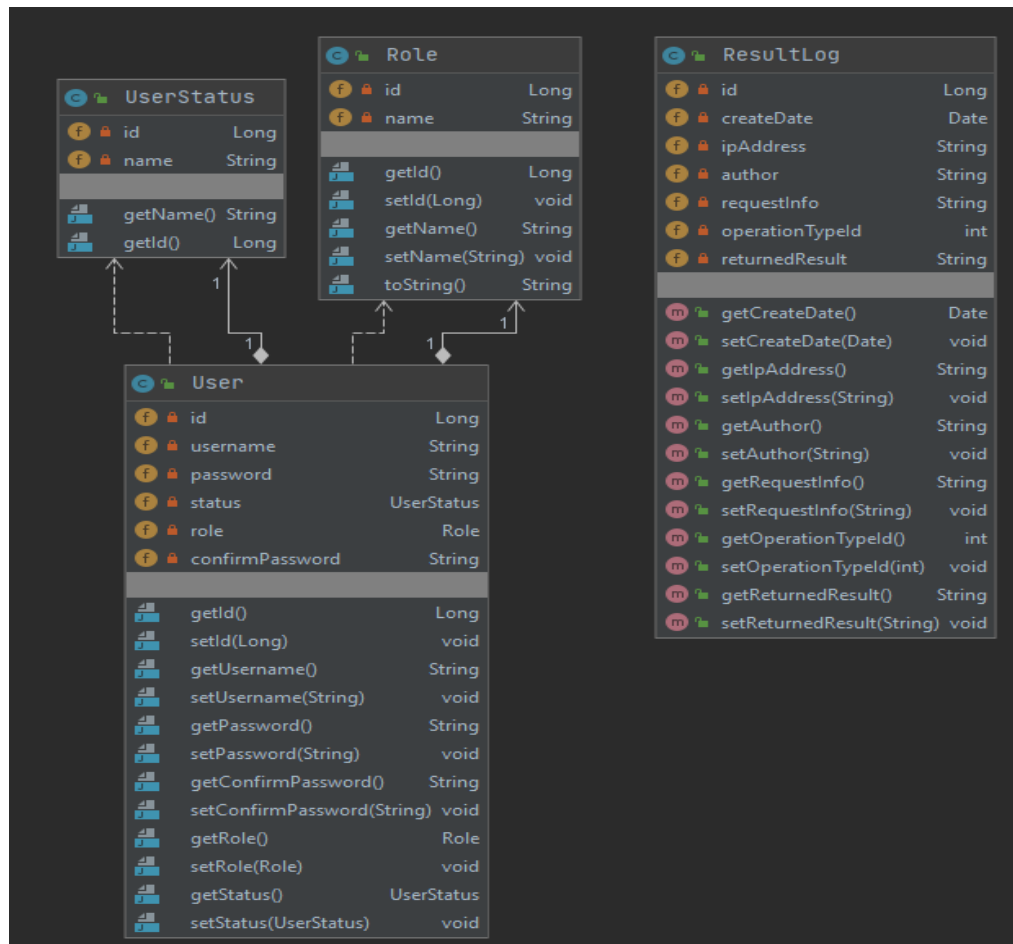


Рисунок 3.10 – Диаграмма классов моделей

На рисунке 3.11 изображена классовая диаграмма контроллеров и связанных с ними сервисами. Следует обратить внимание, что контроллеры завязываются на интерфейсы сервисов. Это связано с тем, что контроллеры используют IoC контейнеры для получения нужных сервисов.

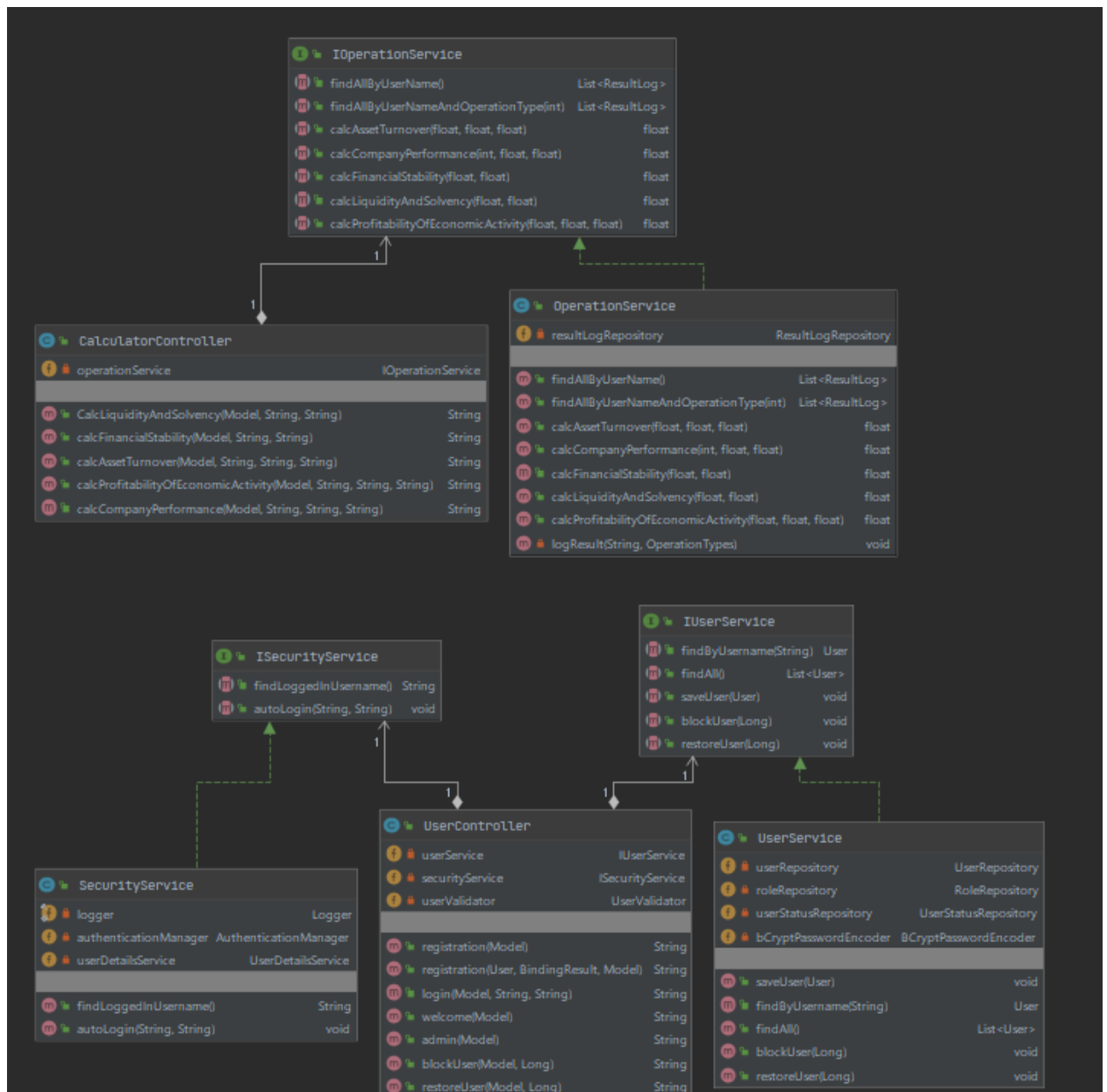


Рисунок 3.10 – Диаграмма классов контроллеров и связанных с ними сервисами

4 Информационная модель системы

Информационной модель разработанной системы состоит из 4 элементов:

- 1) Аудит информация о проведенных операциях (таблица result_logs)
- 2) Пользователи (таблица users)
- 3) Доступные статусы пользователей (таблица user_statuses)
- 4) Доступные роли пользователей (таблица roles)

На рисунке 4.1 представлена диаграмма информационной модели базы данных системы.

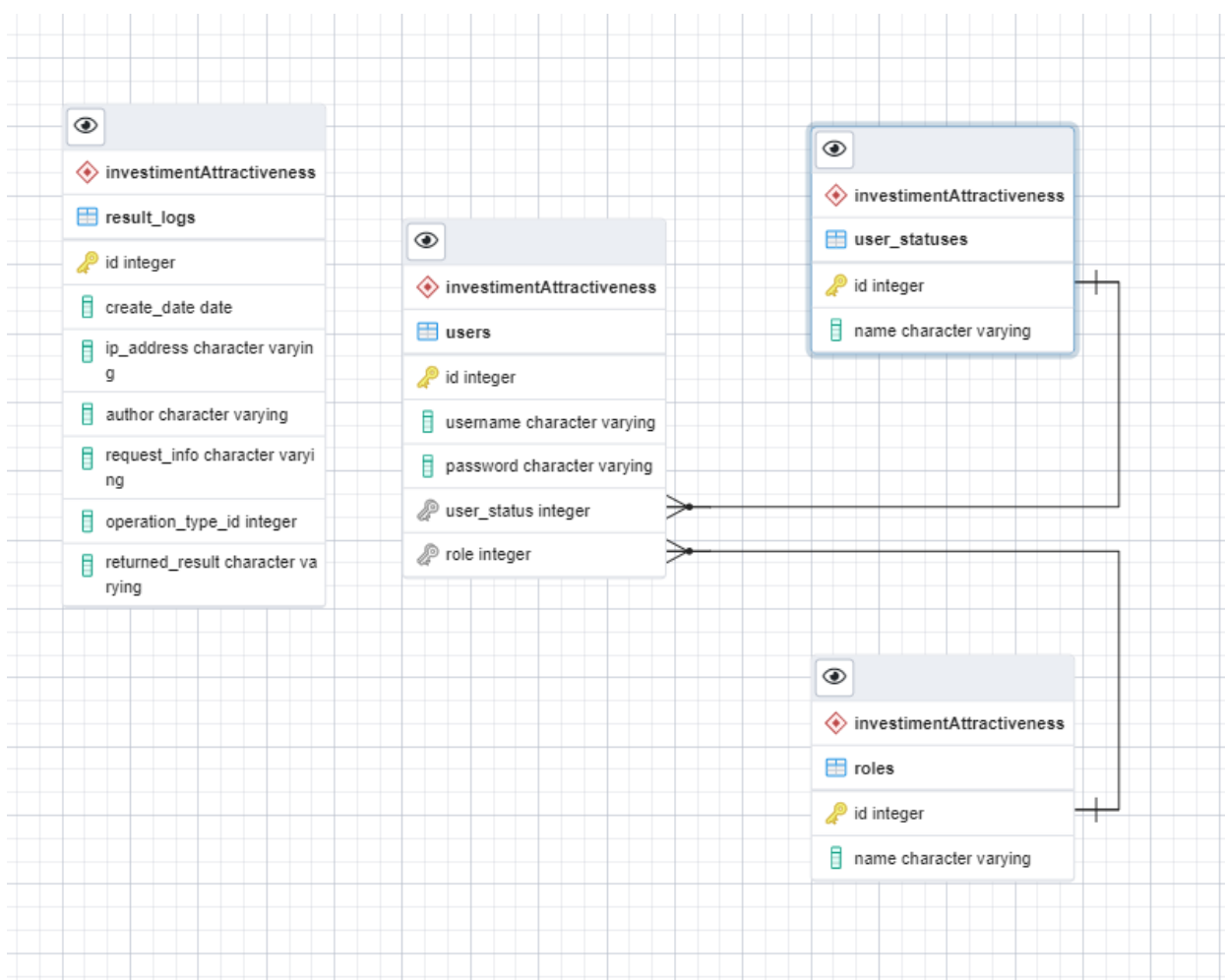


Рисунок 4.1 – Диаграмма информационной модели базы данных

Подобная структура подходит под требования третьей нормальной формы, так как:

- 1) Все атрибуты таблиц атомарны, то есть ни один из атрибутов нельзя разделить на более простые (условие 1 нормальной формы)
- 2) Все не ключевые атрибуты неприводимо зависимы от первичного ключа (условие 2 нормальной формы)
- 3) В таблицах отсутствует транзитивная зависимость, то есть не ключевые столбцы не зависят от значений других не ключевых столбцов (условие 3 нормальной формы) [5]

Исключение составляет таблица `result_logs`. Эта таблица нарушает третью нормальную форму, так как хранит в записях название автора действия. Это связано со спецификой данных, хранящихся в этой таблице. Для аудит информации рекомендуется избавляться от любых зависимостей на сторонние таблицы, так как при удалении связанных сущностей будет нарушаться ссылочная целостность, а в более общем случае – СУБД не даст удалить такую запись, при условии, что не заданы правила каскадного удаления.

5 Обоснование оригинальных решений по использованию технических и программных средств, не включенных в требования

При разработке системы была использована библиотека Hibernate, которая упрощает работу с базами данных. Библиотека помогает разработчику избавиться от написания низкоуровневого кода при работе с БД. Также, при помощи специальных аннотаций, позволяет легко соотнести модели приложения с существующими таблицами. Также Hibernate предоставляет разработчику автоматическую генерацию набора таблиц, построение запросов и обработку полученных данных.

Также была использована спецификация JPA (Java persistence API), которая упрощает в разы написание кода для получения данных из базы данных. Данный интерфейс использует Hibernate для составления запросов в БД. Благодаря данной спецификации, разработчик при помощи специального синтаксиса и аннотаций может реализовывать обращения в БД без написания логики обращения.

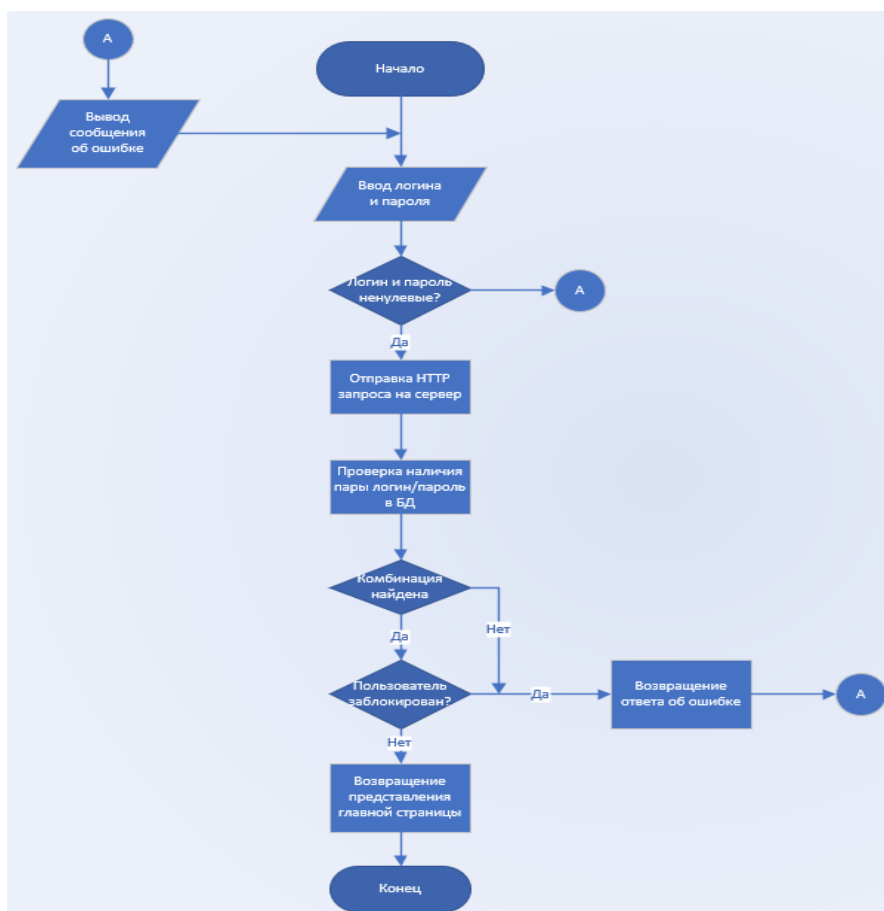
Для упрощения написания клиентской части приложения была использована библиотека Bootstrap, содержащая уже готовые решения по написанию CSS кода. Bootstrap позволяет сэкономить время разработки клиентской части и легок в освоении.

6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы

3.1 Алгоритм процесса аутентификации

Алгоритм начинается с ввода логина и пароля пользователем. Если пользователь оставит поля с логином и паролем пустыми, то выведется сообщение с ошибкой. Если ввод корректный, то клиентский код формирует запрос на сервер. На сервере идет проверка наличия пары логина и пароля в базе данных и, если комбинация не найдена, возвращается ответ с ошибкой. В случае, если пользователь был найден в БД, приложение проверяет статус пользователя и, если он заблокирован, то также сервер возвращает ответ с ошибкой. Если же все проверки были пройдены, пользователю возвращается представление главной страницы.

Блок-схема алгоритма представлена на рисунке 6.1



3.2 Алгоритм получения одного из показателей

Алгоритм начинается с перехода на форму ввода логина и пароля. После того, как данные были введены, идет процесс аутентификации и, если пользователь не найден, то алгоритм возвращается в самое начало. Если пользователь найден, то далее идет переход на страницу подсчета показателя.

При переходе на страницу формируется HTTP Get запрос на получение представления страницы. При получении запроса, сервер проверяет роли текущего юзера и, если в объекте Authority нет роли ROLE_USER, то пользователю возвращается ответ, перенаправляющий его на страницу логина. Если у пользователя есть доступ к ресурсу, то ему предоставляется страница с подсчетом показателя.

После ввода годовых показателей компании и нажатия кнопки «посчитать», клиент отправляет запрос на высчитывание показателя. На этапе получения запроса, сервер также проверяет на наличие роли ROLE_USER в объекте Authority. Далее, идет проверка введенных данных и, если данные валидные, высчитывается значение показателя. Затем, система сохраняет полученный результат в таблицу с аудит данными. После этого, пользователю возвращается представление с уже посчитанными результатами.

Блок-схема алгоритма представлена на рисунке 6.2.

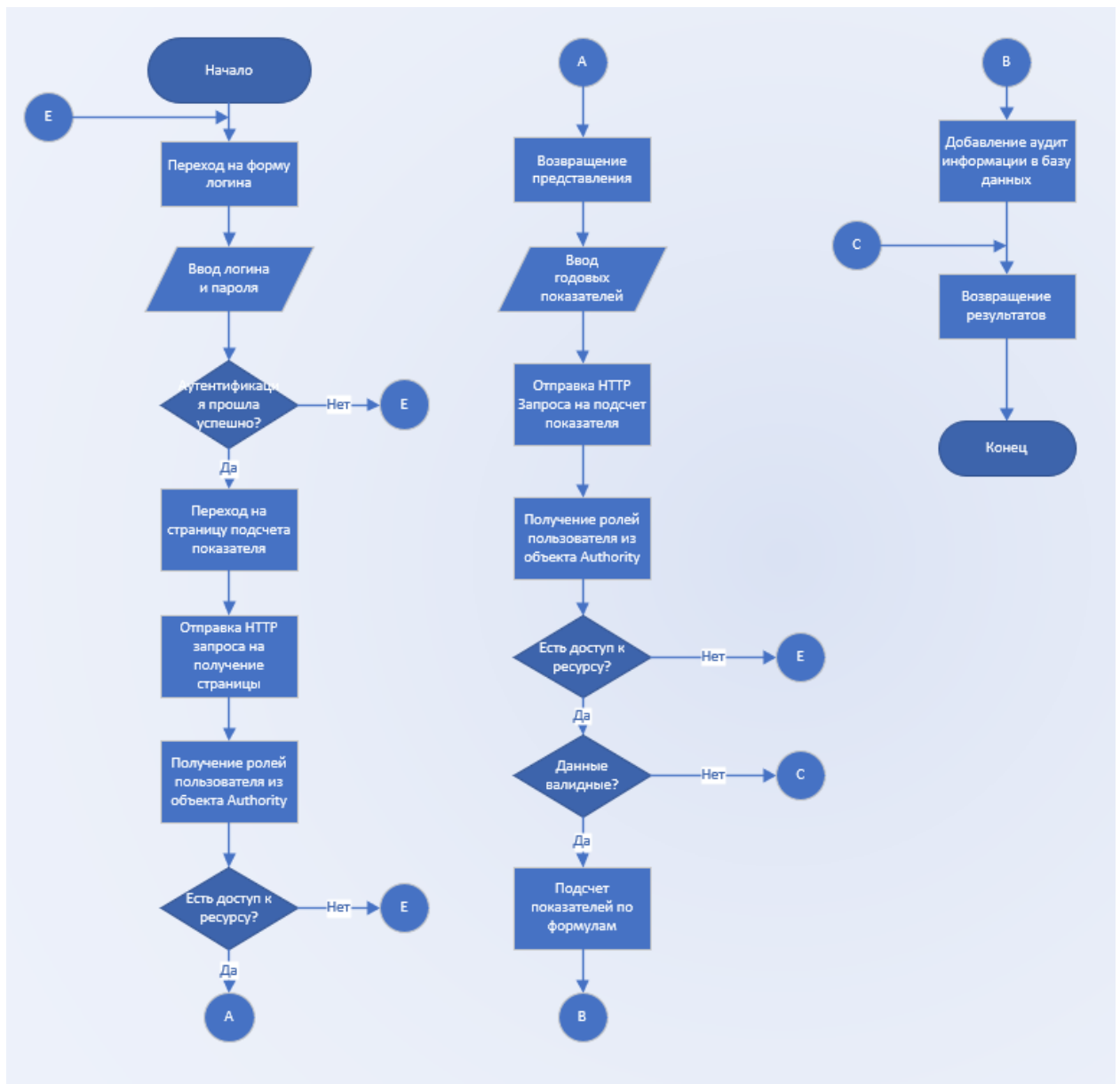


Рисунок 6.2 – Блок-схема процесса получения показателя

7 Руководство пользователя

7.1 Руководство по разворачиванию проекта

Для запуска приложения на локальном сервере необходимо установить заранее:

- JDK 9
- Maven
- PostgreSQL 9
- Tomcat 8.5.*

Алгоритм разворачивания, следующий:

- 1) (Опционально) Откройте командную строку в папке с проектом и сгенерируйте war файл при помощи Maven, вызвав команду `mvn package`
- 2) Создайте схему базы данных используя скрипт `db_create.sql` из папки с проектом
- 3) Запустите сервер Tomcat. Это можно сделать при помощи файла `bin/startup.bat` из папки Tomcat.
- 4) Перейдите на по адресу <http://localhost:8080/manager/> (порт может быть другим, если был изменен в конфигурационном файле)
- 5) Зайдите под пользователем `admin/admin` (может быть изменено в конфигурационном файле)
- 6) В секции WAR file to deploy выберите сгенерированный на первом шаге файл, рисунок 7.1.

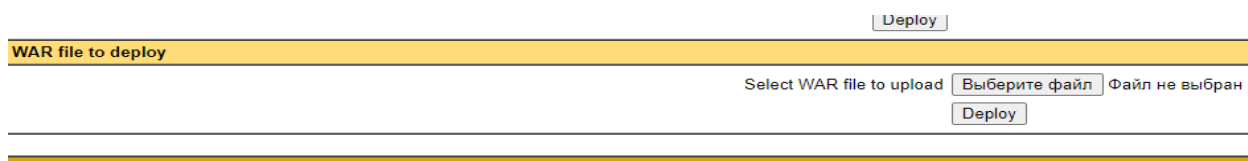


Рисунок 7.1 – Секция с загрузкой .war файла

7) После загрузки файла, в секции Applications должно появиться приложение в статусе Running = true, рисунок 7.2.

Applications			
Path	Version	Display Name	Running
L	None specified	investmentAttractivenessAnalyzer	true

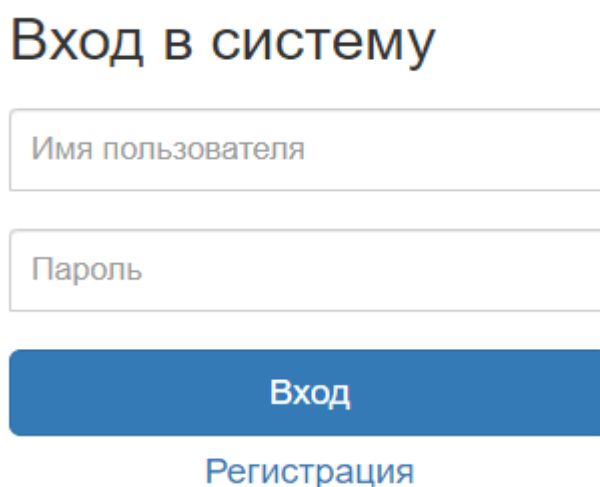
Рисунок 7.2 – Результат успешного разворачивания проекта

8) Теперь можно перейти на сайт нажав на ссылку в Path.

Обратите внимание, что скрипт создания базы данных не предусматривает создание пользователей. Поэтому, для создания администратора необходимо зарегистрироваться вручную и проставить Role = 2 в БД в таблице пользователей.

7.2 Руководство по использованию проекта

При переходе на сайт, пользователю открывается окно входа (рисунок 7.3).



Вход в систему

Имя пользователя

Пароль

Вход

Регистрация

Рисунок 7.3 – Окно входа

Если пользователь впервые заходит на сайт, ему необходимо зарегистрироваться, нажав на соответствующую кнопку.

После перехода по кнопке «Регистрация» открывается соответствующее окно (рисунок 7.4), где пользователю предлагается ввести логин и пароль.

Регистрация

Имя пользователя

Пароль

Подтвердите пароль

Зарегистрироваться

Рисунок 7.4 – Окно регистрации

После регистрации, либо после ввода корректных логина и пароля на странице входа, пользователю открывается главная страница приложения

Домой Ликвидность и платежеспособность Финансовая способность Оборачиваемость активов Рентабельность хозяйственной деятельности Инвестиционная привлекательность Test1 Выход

Система анализа инвестиционной привлекательности

Расчитать показатели ликвидности и платежеспособности

Расчитать показатели финансовой устойчивости

Расчитать показатели оборачиваемости активов

Расчитать показатели рентабельности хозяйственной деятельности

Расчитать интегральный показатель инвестиционной привлекательности компании-эмитента ценных бумаг

Рисунок 7.5 – Главная страница пользователя

Здесь, перейдя по соответствующим кнопкам в центре экрана, либо по навигационной панели сверху, пользователь может перейти на страницу подсчета нужного показателя.

Перейдя по кнопке, пользователю представляется страница, где можно ввести нужные для расчета данные и получить результат (рисунок 7.6).

Показатели рентабельности хозяйственной деятельности

Наиболее важную роль из данной группы показателей в оценке инвестиционной привлекательности организации-эмитента ценных бумаг играет показатель рентабельности активов (капитала). Рентабельность активов комплексно характеризует эффективность деятельности экономического субъекта. При помощи данного показателя можно оценить эффективность управления, поскольку получение высокой прибыли и достаточного уровня доходности во многом зависит от правильности выбора и рациональности принимаемых управленческих решений. По значению уровня рентабельности капитала можно оценить долгосрочное благополучие компании, то есть ее способность получать ожидаемую норму прибыли на инвестиции в достаточно длительной перспективе. При определении рентабельности активов следует учитывать тот факт, что численное значение стоимости имущества не остается неизменным за период по причине ввода в эксплуатацию новых основных средств или их выбытия. Поэтому при исчислении рентабельности активов следует определять их среднее значение. Наиболее правильным при этом является расчет средней хронологической величины инвестированного капитала, а при отсутствии или недостаточности данных можно использовать средние арифметические значения.

Расчет показателей

Чистая прибыль, тыс. руб.

Стоимость имущества компании на начало года, тыс. руб.

Стоимость имущества компании на конец года, тыс. руб.

Рассчитать

Рисунок 7.6 – Страница подсчета показателя рентабельности хозяйственной деятельности

После ввода данных и нажатия на кнопку «Рассчитать», пользователю выводятся результаты (рисунок 7.7).

Результат: 0.044444446

Введенные данные:

Чистая прибыль, тыс. руб. - 1000

Стоимость имущества компании на начало года, тыс. руб. - 25000

Стоимость имущества компании на конец года, тыс. руб. - 20000

Рисунок 7.7 – Результаты подсчета

Для администраторов системы доступна дополнительная функциональность. На главной странице приложения администратор может перейти в раздел администрирования через кнопку «Управление пользователями» (рисунок 7.8).

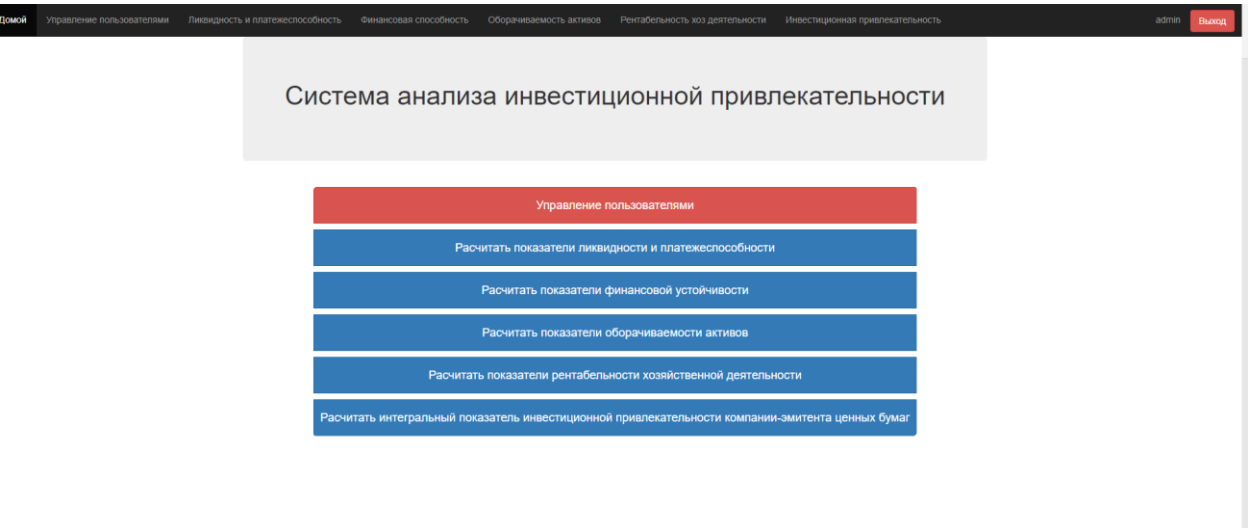


Рисунок 7.8 – Главная страница администратора

Перейдя по кнопке, администратору открывается таблица позволяющая просматривать пользователей системы и управлять доступом к их учетной записи (рисунок 7.9).

Пользователи системы

Идентификатор	Имя	Роль	Статус	
3	admin	ROLE_ADMIN	Активен	
4	username	ROLE_USER	Заблокирован	Разблокировать
5	username1	ROLE_USER	Заблокирован	Разблокировать
6	Test1	ROLE_USER	Активен	Заблокировать

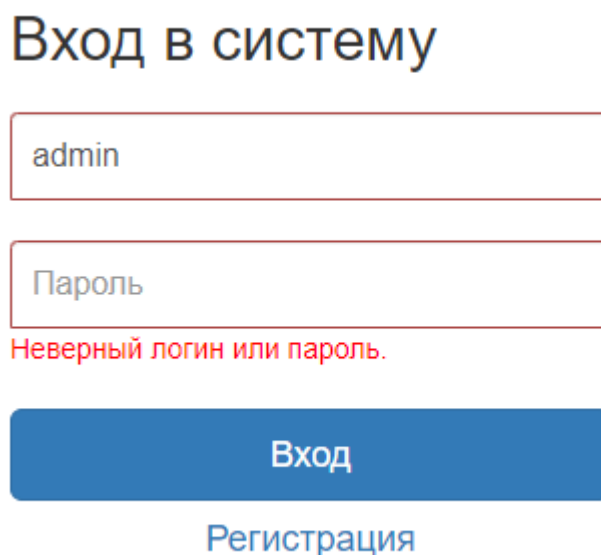
Рисунок 7.9 – Страница администрирования пользователей

8 Результаты тестирования разработанной системы

Разработанная система поддерживает обработку исключительных ситуаций и адекватно реагирует на некорректный ввод данных.

8.1 Тестирование модуля входа в систему

При вводе несуществующих данных в окне входа, пользователю выдается соответствующее сообщение (рисунок 8.1).



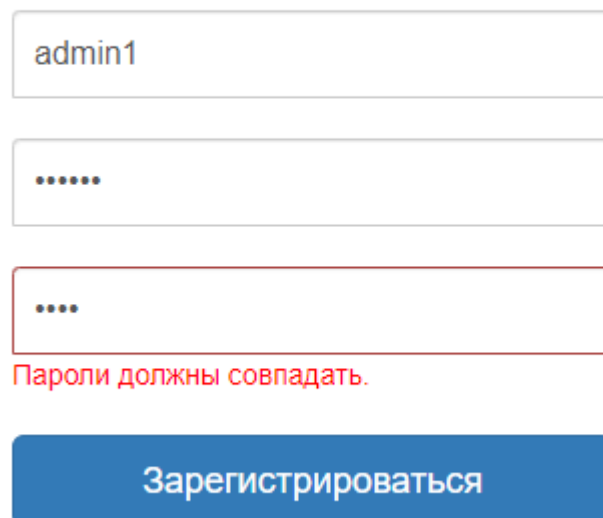
The screenshot shows a login interface titled "Вход в систему". It contains two input fields: the first contains the text "admin", and the second is labeled "Пароль". Below the password field, a red error message reads "Неверный логин или пароль.". At the bottom, there is a blue button labeled "Вход" and a blue link labeled "Регистрация".

Рисунок 8.1 – Ввод несуществующего логина или пароля

8.2 Тестирование модуля регистрации

В окне регистрации предусмотрено 4 проверки: совпадение подтверждения пароля с паролем (рисунок 8.2), проверка на существование имени пользователя (8.3) и проверка корректности данных (рисунок 8.4).

Регистрация



admin1

.....

....

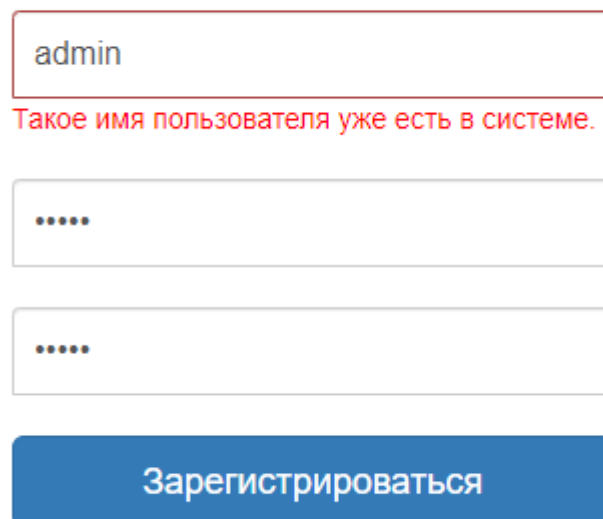
Пароли должны совпадать.

Зарегистрироваться

The registration form is titled 'Регистрация'. It contains three input fields: a username field with 'admin1', a password field with '.....', and a confirmation password field with '....'. A red error message 'Пароли должны совпадать.' (Passwords must match.) is displayed below the confirmation password field. At the bottom is a blue button labeled 'Зарегистрироваться' (Register).

Рисунок 8.2 – Проверка на совпадение паролей

Регистрация



admin

Такое имя пользователя уже есть в системе.

.....

.....

Зарегистрироваться

The registration form is titled 'Регистрация'. It contains three input fields: a username field with 'admin', a password field with '.....', and a confirmation password field with '.....'. A red error message 'Такое имя пользователя уже есть в системе.' (This username already exists in the system.) is displayed below the username field. At the bottom is a blue button labeled 'Зарегистрироваться' (Register).

Рисунок 8.3 – Проверка на существование имени пользователя

Регистрация

admin1

Пароль

Это поле обязательное.
Пароль должен быть от 5 до 32 символов.

Подтвердите пароль

Зарегистрироваться

Рисунок 8.4 – Проверка корректности данных

8.3 Тестирование модуля подсчета показателей

Для этого модуля, в целом, предусмотрен только один вид валидации – проверка на корректность данных для формул (рисунок 8.5).

$$Ктл = \frac{ОА}{КО}$$

где Ктл — коэффициент текущей ликвидности;
ОА — оборотные активы, тыс. руб.;
КО — краткосрочные обязательства, тыс. руб.

Расчет показателей

Оборотные активы, тыс. руб.

211

Краткосрочные обязательства, тыс. руб.

0

Рассчитать

Ошибка: Параметр в знаменателе не может быть равен нулю

Рисунок 8.5 – Проверка деления на ноль

Заключение

По результатам курсовой работы была создана система анализа инвестиционной привлекательности организации-эмитента. Во время разработки программы были учтены все требования, предъявляемые к приложению такого рода.

Подобная система может помочь владельцам компаний проанализировать показатели своей организации и сделать соответствующие выводы.

В результате написания работы были изучены принципы разработки веб-приложения на Java с использованием фреймворка Spring, изучена база данных PostgreSQL и были закреплены общие знания по UML-проектированию.

Данная курсовая работа является лишь первой версией разрабатываемой системы. В дальнейшем планируется добавление нескольких новых модулей, таких как: сохранение данных об организациях, предоставление отчетных графиков, анализ компании во временном промежутке.

Список использованных источников

- [1] Сайт Электро-2021 [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.elektro-expo.ru/ru/articles/avtomatizaciya-sistem-upravleniya/>
- [2] Сайт Центр управления финансами [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://center-yf.ru/data/ip/investicionnaya-privlekatelnost.php>
- [3] Сайт Корпоративный менеджмент [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.cfin.ru/finanalysis/invest/issuing_company.shtml
- [4] Сайт БГУИР [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.bsuir.by/m/12_100229_1_65759.pdf
- [5] Сайт Хабр [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habr.com/ru/post/129195/>

Приложение А

Листинг скрипта генерации базы данных

```
create schema "investmentAttractiveness";
alter schema "investmentAttractiveness" owner to postgres;
create table if not exists result_logs
(
    id integer generated always as identity constraint result_logs_pkey
primary key,
    create_date date not null,
    ip_address varchar not null,
    author varchar not null,
    request_info varchar not null,
    operation_type_id integer not null,
    returned_result varchar
);

alter table result_logs owner to postgres;
create table if not exists roles
(
    id integer not null constraint roles_pkey primary key,
    name varchar not null
);

alter table roles owner to postgres;
create table if not exists user_statuses
(
    id integer not null constraint user_statuses_pkey primary key,
    name varchar not null
);

alter table user_statuses owner to postgres;

create table if not exists users
(
    id integer generated always as identity constraint users_pkey primary
key,
    username varchar not null,
    password varchar not null,
    user_status integer not null constraint status_fk references
user_statuses,
    role integer not null constraint role_fk references roles
);

alter table users owner to postgres;

INSERT INTO "investmentAttractiveness".roles(id, name)
VALUES (1, 'ROLE_USER'),
       (2, 'ROLE_ADMIN');

INSERT INTO "investmentAttractiveness".user_statuses(id, name)
VALUES (1, 'Активен'),
       (2, 'Заблокирован');
```

Приложение Б

Листинг основных элементов программы

UserController

```
@Controller
public class UserController {

    @Autowired
    private IUserService userService;

    @Autowired
    private ISecurityService securityService;

    @Autowired
    private UserValidator userValidator;

    @RequestMapping(value = "/registration", method = RequestMethod.GET)
    public String registration(Model model) {
        model.addAttribute("userForm", new User());

        return "registration";
    }

    @RequestMapping(value = "/registration", method = RequestMethod.POST)
    public String registration(@ModelAttribute("userForm") User userForm,
        BindingResult bindingResult, Model model) {
        userValidator.validate(userForm, bindingResult);

        if (bindingResult.hasErrors()) {
            return "registration";
        }

        userService.saveUser(userForm);
        securityService.autoLogin(userForm.getUsername(),
            userForm.getConfirmPassword());

        return "main";
    }

    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public String login(Model model, String error, String logout) {
        if (error != null) {
            model.addAttribute("error", "Неверный логин или пароль.");
        }

        if (logout != null) {
            model.addAttribute("message", "Вы вышли из системы.");
        }

        return "login";
    }

    @PreAuthorize("hasAnyRole('ROLE_USER', 'ROLE_ADMIN')")
    @RequestMapping(value = {"/", "/welcome"}, method = RequestMethod.GET)
    public String welcome(Model model) {
        return "main";
    }
}
```

```

    }

    @PreAuthorize("hasRole('ROLE_ADMIN')")
    @RequestMapping(value = "/admin", method = RequestMethod.GET)
    public String admin(Model model) {
        model.addAttribute("users", userService.findAll());

        return "admin";
    }

    @PreAuthorize("hasRole('ROLE_ADMIN')")
    @RequestMapping(value = "/blockUser", method = RequestMethod.GET)
    public String blockUser(Model model, @RequestParam(value = "id") final
Long id) {
        userService.blockUser(id);
        model.addAttribute("users", userService.findAll());

        return "admin";
    }

    @PreAuthorize("hasRole('ROLE_ADMIN')")
    @RequestMapping(value = "/restoreUser", method = RequestMethod.GET)
    public String restoreUser(Model model, @RequestParam(value = "id") final
Long id) {
        userService.restoreUser(id);
        model.addAttribute("users", userService.findAll());

        return "admin";
    }
}

```

CalculatorController

```

@Controller
public class CalculatorController {

    @Autowired
    private IOperationService operationService;

    @PreAuthorize("hasAnyRole('ROLE_USER','ROLE_ADMIN')")
    @RequestMapping(value = {"calculator/liquidity-and-solvency"}, method =
RequestMethod.GET)
    public String CalcLiquidityAndSolvency(Model model,
                                           @RequestParam(value = "OA", required = false)
final String OA,
                                           @RequestParam(value = "KO", required = false)
final String KO) {
        float result = 0;
        if (OA != null && KO != null && !OA.equals("") && !KO.equals("")) {
            if(Float.parseFloat(KO) == 0) {
                model.addAttribute("error", "Параметр в знаменателе не может
быть равен нулю");
            } else {
                model.addAttribute("error", "");
                result =
operationService.calcLiquidityAndSolvency(Float.parseFloat(KO),
Float.parseFloat(OA));
            }

            model.addAttribute("result", result);
        }
    }
}

```



```

        model.addAttribute("allResult",
operationService.findAllByUserNameAndOperationType(OperationTypes.LiquidityAn
dSolvency.getOperationType()));

        model.addAttribute("value1", OA);
        model.addAttribute("value2", KO);

        return "calculator/liquidityAndSolvencyCalc";
    }

    @PreAuthorize("hasAnyRole('ROLE_USER','ROLE_ADMIN')")
    @RequestMapping(value = {"calculator/financial-stability"}, method =
RequestMethod.GET)
    public String calcFinancialStability(Model model,
        @RequestParam(value = "SK", required = false)
final String SK,
        @RequestParam(value = "VB", required = false)
final String VB) {
        float result = 0;
        if (SK != null && VB != null && !SK.equals("") && !VB.equals("")) {
            if(Float.parseFloat(VB) == 0) {
                model.addAttribute("error", "Параметр в знаменателе не может
быть равен нулю");
            } else {
                model.addAttribute("error", "");
                result =
operationService.calcFinancialStability(Float.parseFloat(SK),
Float.parseFloat(VB));
            }
        }

        model.addAttribute("result", result);
        model.addAttribute("allResult",
operationService.findAllByUserNameAndOperationType(OperationTypes.FinancialSt
ability.getOperationType()));

        model.addAttribute("value1", SK);
        model.addAttribute("value2", VB);

        return "calculator/financialStabilityCalc";
    }

    @PreAuthorize("hasAnyRole('ROLE_USER','ROLE_ADMIN')")
    @RequestMapping(value = {"calculator/asset-turnover"}, method =
RequestMethod.GET)
    public String calcAssetTurnover(Model model,
        @RequestParam(value = "N", required = false) final
String N,
        @RequestParam(value = "SANG", required = false)
final String SANG,
        @RequestParam(value = "SAKG", required = false)
final String SAKG) {
        float result = 0;
        if (N != null && SANG != null && SAKG != null && !N.equals("") &&
!SANG.equals("") && !SAKG.equals("")){
            result =
operationService.calcAssetTurnover(Float.parseFloat(SANG),
Float.parseFloat(SAKG), Float.parseFloat(N));
        }

        model.addAttribute("result", result);
        model.addAttribute("allResult",
operationService.findAllByUserNameAndOperationType(OperationTypes.AssetTurnov

```

```

er.getOperationType());

    model.addAttribute("value1", N);
    model.addAttribute("value2", SANG);
    model.addAttribute("value3", SAKG);

    return "calculator/assetTurnoverCalc";
}

@PreAuthorize("hasAnyRole('ROLE_USER','ROLE_ADMIN')")
@RequestMapping(value = {"calculator/profitability-of-economic-activity"}, method = RequestMethod.GET)
public String calcProfitabilityOfEconomicActivity(Model model,
    @RequestParam(value = "RC", required = false) final
String RC,
    @RequestParam(value = "AKG", required = false) final
String AKG,
    @RequestParam(value = "ANG", required = false) final
String ANG) {
    float result = 0;
    if (RC != null && AKG != null && ANG != null && !RC.equals("") &&
!AKG.equals("") && !ANG.equals("")) {
        result =
operationService.calcProfitabilityOfEconomicActivity(Float.parseFloat(RC),
Float.parseFloat(AKG), Float.parseFloat(ANG));
    }

    model.addAttribute("result", result);
    model.addAttribute("allResult",
operationService.findAllByUsernameAndOperationType(OperationTypes.Profitabili
tyOfEconomicActivity.getOperationType()));

    model.addAttribute("value1", RC);
    model.addAttribute("value2", AKG);
    model.addAttribute("value3", ANG);

    return "calculator/profitabilityOfEconomicActivityCalc";
}

@PreAuthorize("hasAnyRole('ROLE_USER','ROLE_ADMIN')")
@RequestMapping(value = {"calculator/company-performance"}, method =
RequestMethod.GET)
public String calcCompanyPerformance(Model model,
    @RequestParam(value = "N", required = false) final
String N,
    @RequestParam(value = "XIJE", required = false) final
String XIEJ,
    @RequestParam(value = "XIET", required = false)
final String XIET) {
    float result = 0;

    if (N != null && XIEJ != null && XIET != null && !N.equals("") &&
!XIEJ.equals("") && !XIET.equals("")) {
        if (Float.parseFloat(XIET) == 0) {
            model.addAttribute("error", "Параметр в знаменателе не может
быть равен нулю");
        } else {
            model.addAttribute("error", "");
            result =
operationService.calcCompanyPerformance(Integer.valueOf(N),
Float.parseFloat(XIEJ), Float.parseFloat(XIET));
        }
    }
}

```

```

        model.addAttribute("result", result);
        model.addAttribute("allResult",
operationService.findAllByUserNameAndOperationType(OperationTypes.CompanyPerformance.getOperationType()));

        model.addAttribute("value1", N);
        model.addAttribute("value2", XIEJ);
        model.addAttribute("value3", XIET);

        return "calculator/companyPerformanceCalc";
    }
}

```

UserService

```

@Service
public class UserService implements IUserService {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleRepository roleRepository;

    @Autowired
    private UserStatusRepository userStatusRepository;

    @Autowired
    private BCryptPasswordEncoder bCryptPasswordEncoder;

    @Override
    public void saveUser(User user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        user.setRole(roleRepository.findOne((long) Roles.User.getRole()));
        user.setStatus(userStatusRepository.findOne((long)
UserStatuses.Active.getStatus()));
        userRepository.save(user);
    }

    @Override
    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    @Override
    public List<User> findAll() {
        return userRepository.findAll();
    }

    @Override
    public void blockUser(Long id) {
        User user = userRepository.findById(id);
        user.setStatus(userStatusRepository.findOne((long)
UserStatuses.Blocked.getStatus()));
        userRepository.save(user);
    }

    @Override
    public void restoreUser(Long id) {
        User user = userRepository.findById(id);
    }
}

```

```

        user.setStatus(userStatusRepository.getOne((long)
UserStatuses.Active.getStatus()));
        userRepository.save(user);
    }
}

```

OperationService

```

@Service
public class OperationService implements IOperationService {

    @Autowired
    private ResultLogRepository resultLogRepository;

    @Override
    public List<ResultLog> findAllByUserName() {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        return resultLogRepository.getAllByAuthor(authentication.getName());
    }

    @Override
    public List<ResultLog> findAllByUserNameAndOperationType(int
operationType) {
        Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

        return
resultLogRepository.getAllByAuthorAndOperationTypeId(authentication.getName()
, operationType);
    }

    @Override
    public float calcAssetTurnover(float SANG, float SAKG, float N) {
        float result = N / ((SANG + SAKG) / 2);
        logResult(String.valueOf(result), OperationTypes.AssetTurnover);

        return result;
    }

    @Override
    public float calcCompanyPerformance(int N, float XIEJ, float XIET) {
        double kipP = (1 - (XIEJ / XIET));
        for (int i = 1; i >= N; i++) {
            kipP = pow(kipP + 1, 2);
        }

        float result = (float) sqrt(kipP);
        logResult(String.valueOf(result), OperationTypes.CompanyPerformance);

        return result;
    }

    @Override
    public float calcFinancialStability(float SK, float VB) {
        float result = SK/VB;
        logResult(String.valueOf(result), OperationTypes.FinancialStability);

        return result;
    }
}

```

```

@Override
public float calcLiquidityAndSolvency(float KO, float OA) {
    float result = OA/KO;
    logResult(String.valueOf(result),
OperationTypes.LiquidityAndSolvency);

    return result;
}

@Override
public float calcProfitabilityOfEconomicActivity(float RC, float AKG,
float ANG) {
    float result = RC / ((AKG + ANG) / 2);
    logResult(String.valueOf(result),
OperationTypes.ProfitabilityOfEconomicActivity);

    return result;
}

private void logResult(String result, OperationTypes operationType) {
    Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();

    ResultLog resultLog = new ResultLog();
    resultLog.setAuthor(authentication.getName());
    resultLog.setOperationTypeId(operationType.ordinal());
    resultLog.setReturnedResult(result);
    resultLog.setCreateDate(new Date());

    resultLog.setIpAddress(HttpRequestUtils.getRemoteIP(RequestContextHolder.curr
entRequestAttributes()));
    HttpServletRequest currentRequest =
HttpRequestUtils.getCurrentHttpRequest();
    Map queryParams = currentRequest.getParameterMap();
    Gson gson = new Gson();
    resultLog.setRequestInfo(gson.toJson(queryParams));

    resultLogRepository.save(resultLog);
}
}

```

web.xml

```

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">

    <display-name>investmentAttractivenessAnalyzer</display-name>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/appconfig-root.xml</param-value>
    </context-param>

    <filter>
        <filter-name>CharsetFilter</filter-name>
        <filter-
class>net.investmentAttractivenessAnalyzer.filters.EncodingFilter</filter-

```

```

class>
    <init-param>
        <param-name>requestEncoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>CharsetFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value></param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
</web-app>

```

appconfig-root.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd

    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <import resource="appconfig-mvc.xml"/>

    <import resource="appconfig-data.xml"/>

    <import resource="appconfig-security.xml"/>

```

```

    <context:component-scan base-
package="net.investmentAttractivenessAnalyzer.*"/>

    <context:property-placeholder location="classpath:database.properties"/>

</beans>

```

appconfig-data.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:jpa="http://www.springframework.org/schema/data/jpa"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns="http://www.springframework.org/schema/beans"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/data/jpa
http://www.springframework.org/schema/data/jpa/spring-jpa.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx.xsd">

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="packagesToScan"
value="net.investmentAttractivenessAnalyzer.models"/>
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"/>
        </property>
        <property name="jpaProperties">
            <props>
                <prop
key="hibernate.dialect">org.hibernate.dialect.PostgreSQL9Dialect</prop>
                <prop
key="hibernate.default_schema">"investmentAttractiveness"</prop>
                <prop key="hibernate.show_sql">true</prop>
            </props>
        </property>
    </bean>

    <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="entityManagerFactory"/>
    </bean>

    <tx:annotation-driven/>

    <jpa:repositories base-
package="net.investmentAttractivenessAnalyzer.repositories"/>
</beans>

```

appconfig-mvc.xml

```
<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns="http://www.springframework.org/schema/beans"
       xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <mvc:annotation-driven/>

    <mvc:resources mapping="/resources/**" location="/resources/" />

    <bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSou
rce">
        <property name="basenames">
            <list>
                <value>classpath:validation</value>
            </list>
        </property>
        <property name="defaultEncoding">
            <value>UTF-8</value>
        </property>
    </bean>

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix">
            <value>/WEB-INF/views/</value>
        </property>
        <property name="suffix">
            <value>.jsp</value>
        </property>
    </bean>

</beans>
```

appconfig-security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
             xmlns:beans="http://www.springframework.org/schema/beans"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

    <http auto-config="true">
        <intercept-url pattern="/" access="hasAnyRole('ROLE_USER',
'ROLE_ADMIN')"/>
        <intercept-url pattern="/welcome" access="hasAnyRole('ROLE_USER',
'ROLE_ADMIN')"/>
        <intercept-url pattern="/admin" access="hasRole('ROLE_ADMIN')"/>
        <intercept-url pattern="/blockUser" access="hasRole('ROLE_ADMIN')"/>
        <intercept-url pattern="/restoreUser"
```



```

access="hasRole('ROLE_ADMIN')"/>
    <intercept-url pattern="/calculator/company-performance"
access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')"/>
    <intercept-url pattern="/calculator/profitability-of-economic-
activity" access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')"/>
    <intercept-url pattern="/calculator/asset-turnover"
access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')"/>
    <intercept-url pattern="/calculator/financial-stability"
access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')"/>
    <intercept-url pattern="/calculator/liquidity-and-solvency"
access="hasAnyRole('ROLE_USER', 'ROLE_ADMIN')"/>

    <form-login login-page="/login" default-target-url="/welcome"
authentication-failure-url="/login?error"
        username-parameter="username" password-
parameter="password"/>

    <logout logout-success-url="/login?logout"/>
</http>

<authentication-manager alias="authenticationManager">
    <authentication-provider user-service-ref="userDetailService">
        <password-encoder ref="encoder"></password-encoder>
    </authentication-provider>
</authentication-manager>

<beans:bean id="userDetailService"

class="net.investmentAttractivenessAnalyzer.services.UserDetailService"></bea
ns:bean>

<beans:bean id="encoder"

class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder">
    <beans:constructor-arg name="strength" value="11"/>
</beans:bean>
</beans:beans>

```