

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

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Системный анализ и проектирование информационных
систем (с разделом Разработка веб-приложений)»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
Старший преподаватель
_____ Т.В. Русак
____.____.2023

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему:
**«ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО
СРЕДСТВА ПРОГНОЗИРОВАНИЯ ДИВИДЕНДНОЙ ДОХОДНОСТИ
ЦЕННЫХ БУМАГ»**

БГУИР КР 1-40 05 01-08 85 ПЗ

Выполнил студент группы 173601
ЕПИХОВ Натан Сергеевич

(подпись студента)
Курсовая работа представлена на
проверку _____.____.2023

(подпись студента)

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА	5
1.1 Описание предметной области	5
1.2 Разработка функциональной модели предметной области	7
1.3 Анализ требований к разрабатываемому программному средству	8
1.4 Разработка информационной модели предметной области	14
1.5 UML-модели представления программного средства и их описание	16
2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА	20
2.1 Постановка задачи.....	20
2.2 Обоснование выбора компонентов и технологий для реализации программного средства.....	21
2.3 Архитектурные решения	23
2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства	25
2.5 Проектирование пользовательского интерфейса.....	28
2.6 Методы и средства, используемые для обеспечения безопасности данных.....	31
3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА	33
4 РУКОВОДСТВО ПО УСТАНОВКЕ (РАЗВЕРТЫВАНИЮ) И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА.....	35
4.1 Руководство по установке (развертыванию) программного средства	35
4.2 Руководство пользователя.....	37
ЗАКЛЮЧЕНИЕ	42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	43
ПРИЛОЖЕНИЕ А	44
ПРИЛОЖЕНИЕ Б.....	45
ПРИЛОЖЕНИЕ В	58
ПРИЛОЖЕНИЕ Г.....	60
ПРИЛОЖЕНИЕ Д	61
ПРИЛОЖЕНИЕ Е.....	62
ПРИЛОЖЕНИЕ Ж	63
ПРИЛОЖЕНИЕ З.....	64
ПРИЛОЖЕНИЕ И	65

ВВЕДЕНИЕ

В настоящее время трудно представить нашу жизнь без информационных технологий. Компьютеры, ноутбуки, различные технические средства - всё это является неотъемлемой частью существования современного человека и используется ежедневно и в быту, и в работе. Вряд ли можно найти организацию, которая в той или иной степени не была бы оснащена средствами приёма, хранения, передачи информации. Информация является одним из основных ресурсов роста производительности. Эффективное использование информации приобретает всё более важное значение для обеспечения эффективности работы всех сфер существования человека, особенно от эффективного использования информации зависит бизнес. Чем тщательнее для организации подобрано программное обеспечение, тем оно более функционально и экономически эффективно.

Фондовый рынок - это рынок, на котором продажа и покупка акций компаний происходит между инвесторами. Акции представляют собой долю в собственности компании и дают право на получение части прибыли компании и голосование на собраниях акционеров. В данный момент фондовый рынок сильно зависит от средств автоматизации, так как инвесторы зависят от программных средств анализа, мониторинга и моделирования, которые в свою очередь помогают при торговле ценными бумагами.

Ценные бумаги - это финансовые инструменты, которые представляют собой права на определенные обязательства или активы. Это могут быть, например, акции, облигации, паи инвестиционных фондов, фьючерсы и опционы. Акции являются основным видом ценных бумаг, которые торгуются на фондовом рынке. Их цена зависит от многих факторов, включая финансовые результаты компании, ожидания инвесторов относительно ее будущего роста, конкурентную обстановку в отрасли и макроэкономические условия. Облигации также могут быть проданы на фондовом рынке, и они представляют собой заемные обязательства компаний или государств. Облигации генерируют доход в виде процентных выплат и обычно считаются менее рискованными, чем акции.

Однако стоит отметить, что инвестирование в любой отрасли – серьезный риск, так как фондовый и другие рынки ценных бумаг изменяются молниеносно и непредсказуемо. Прежде чем покупать какие-либо ценные бумаги, инвестору необходимо проанализировать ситуацию на рынке.

Целью курсовой работы является проектирование и разработка программного средства прогнозирования дивидендной доходности ценных бумаг, призванная повысить эффективность работы инвестора по анализу рынка ценных бумаг, а также призванная помочь спрогнозировать прибыль от инвестиций.

Для достижения цели необходимо решить следующие задачи:

- выявить и проанализировать существующие программные средства, которые применяются для прогнозирования дивидендной доходности ценных бумаг;
- провести моделирование предметной области;
- определить необходимый функционал разрабатываемого программного средства.
- разработать проект и архитектуру программного обеспечения;
- разработать алгоритмы программного средства выполнить их программную реализацию, описать особенности программной реализации;
- разработать тесты для полученного программного средства, выполнить тестирование, проанализировать полученные результаты.

Для реализации поставленной цели и сформулированных задач данного курсового работы будут использованы следующие инструментальные средства, технологии проектирования разработки:

- редактор диаграмм и блок-схем MS Visio;
- среда выполнения Node.js;
- унифицированный язык моделирования UML;
- язык программирования Java Script;
- система управления базами данных MySQL Server.

В результате проделанной работы будет разработано программное средство для прогнозирования дивидендной доходности ценных бумаг. К основным преимуществам данного программного средства можно отнести такие характеристика как: простота внедрения и пользования, небольшое потребление системных ресурсов, быстродействие, невысокая стоимость.

1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА

1.1 Описание предметной области

Современный фондовый рынок – это глобальный рынок, который представляет собой совокупность различных бирж и торговых площадок, где происходит купля-продажа ценных бумаг компаний. Он является одним из ключевых элементов мировой экономики и имеет огромное значение для инвесторов, компаний и государств. Сегодня на фондовых рынках представлены акции крупнейших компаний мира, включая технологических гигантов, нефтяные компании, фармацевтические компании и другие.

Современный фондовый рынок также характеризуется использованием новых технологий и инструментов, таких как высокочастотная торговля и алгоритмические торговые системы. Это позволяет сделать торговлю на фондовом рынке более быстрой и эффективной, но также создает новые вызовы и проблемы, такие как возможность манипуляций на рынке и сбои в работе систем.

В целом, торговля ценными бумагами является одним из ключевых способов для инвесторов инвестировать свои сбережения и получить прибыль. Кроме того, торговля ценными бумагами также позволяет компаниям привлекать капитал и финансировать свои проекты, которые могут помочь им расти и развиваться.

Для инвесторов, цель торговли ценными бумагами может быть разной. Некоторые инвесторы могут искать долгосрочное инвестирование, ориентируясь на рост компаний и получение дивидендов. Другие инвесторы могут искать быстрый заработок на изменении цен ценных бумаг в короткий промежуток времени, используя различные стратегии и методы технического и фундаментального анализа.

Прогнозирование дивидендной доходности ценных бумаг является одним из важных аспектов анализа инвестиционной привлекательности компаний и их акций. Дивидендная доходность - это отношение дивидендов, выплаченных компанией за определенный период, к текущей цене ее акций.

Важным аспектом прогнозирования дивидендной доходности является понимание того, что это прогноз, а не гарантированная доходность. Даже если компания имеет хорошую историю выплаты дивидендов и положительные финансовые показатели, могут возникнуть факторы, которые могут привести к изменению выплат дивидендов в будущем.

Важно также учитывать, что дивидендная доходность не является единственным критерием при выборе инвестиций в ценные бумаги. Вместе с дивидендами следует учитывать и другие факторы, такие как финансовые показатели компании, перспективы ее роста, конкурентное окружение, рыночные тенденции и так далее.

Но стоит отметить, что хоть дивидендная доходность ценных бумаг и не является единственным верным показателем при выборе активов для инвестирования, но это один из основных критериев выбора, поэтому его анализу необходимо уделить особое внимание при анализе фондового рынка.

Основные бизнес-процессы прогнозирования дивидендной доходности ценных бумаг:

- сбор и анализ финансовой информации;
- анализ исторических данных;
- анализ индустрии и конкурентной среды;
- анализ текущей рыночной ситуации;
- прогнозирование будущих тенденций;
- принятие инвестиционного решения.

Таким образом, прибыльная инвестиция складывается из успешно выполненных вышеперечисленных процессов. В настоящее время на рынке ощущается нехватка программного обеспечения у средних и малых инвесторов, что негативно складывается на проводимом анализе и, как следствие, убытках при инвестировании.

Любая аналитическая деятельность связана с накоплением большого количества информации разного вида. Актуальность хранимой информации, а также своевременность ее получения и доведения до заинтересованных лиц – решающие факторы при принятии верного и взвешенного инвестиционного решения.

1.2 Разработка функциональной модели предметной области

Use-Case диаграмма, также известная как диаграмма прецедентов использования, является графическим представлением функциональности программного обеспечения. Она описывает взаимодействие пользователей с системой и представляет функциональные возможности ПО в виде прецедентов использования. [2]

Use-Case диаграмма состоит из следующих элементов:

- actors - это роли, которые играют пользователи системы;
- use cases - это функциональные возможности системы, которые описываются взаимодействием между ролями и системой;
- связи между ролями и прецедентами - это отношения, которые показывают, какие прецеденты могут быть выполнены ролями.

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Грамотное проектирование Use-Case диаграмм может помочь предотвратить многие ошибки разработки, включая:

- неправильное определение функциональных требований;
- ошибки в процессе сбора и анализа требований;
- ошибки в процессе проектирования;
- ошибки при тестировании;
- неудачный выбор функциональности;
- неэффективное использование ресурсов.

В целом, грамотное проектирование Use-Case диаграмм помогает снизить риски ошибок и увеличить качество разработки программного обеспечения, а также обозначить реализуемый при разработке функционал программного средства.

Диаграмма вариантов использования для пользователя изображена на рисунке 1.1.

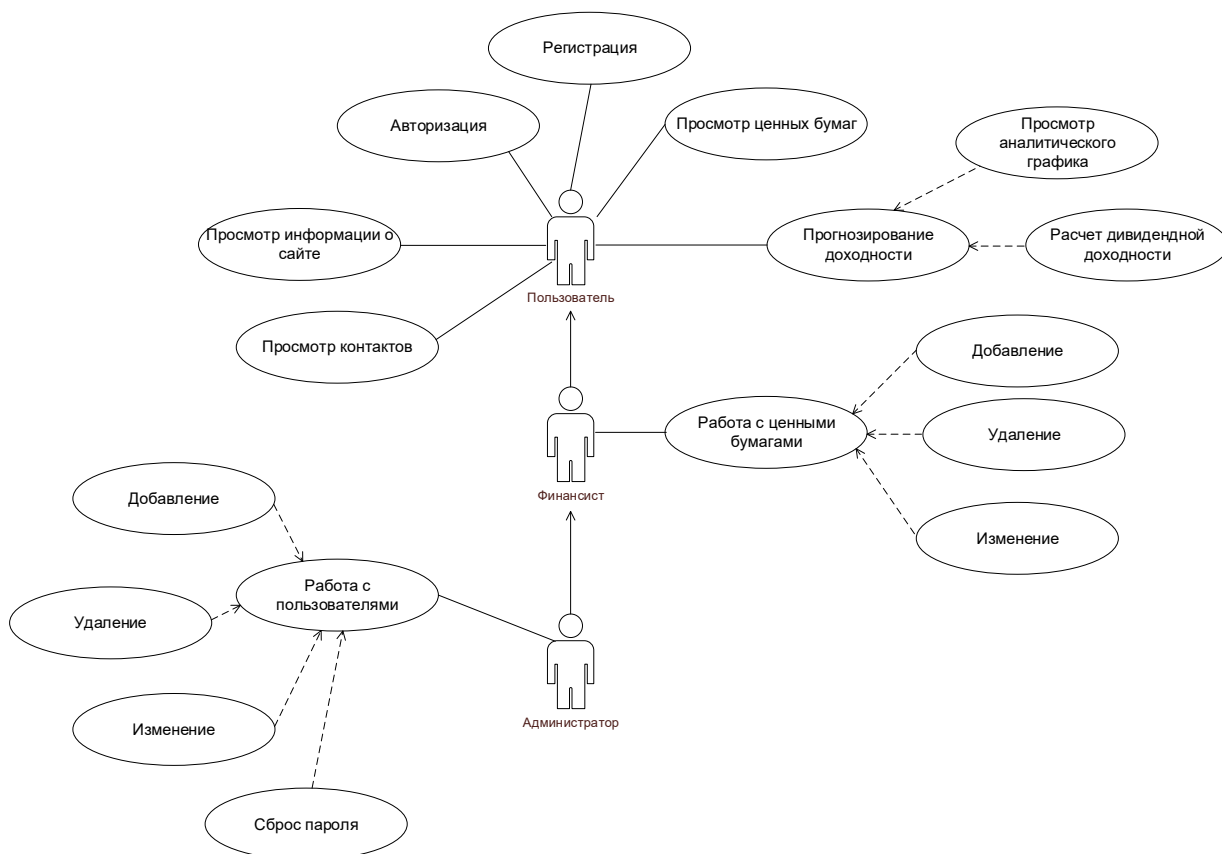


Рисунок 1.1 – Диаграмма вариантов использования системы пользователем

1.3 Анализ требований к разрабатываемому программному средству

Для определения и разработки требований к разрабатываемому программному средству, необходимо осуществить анализ программных средств, существующих на рынке в данный момент.

На данный момент, многие предприятия и частные инвесторы имеют опыт использования различного программного обеспечения. При анализе современного рынка программных продуктов было выделено несколько популярных программ, обеспечивающих прогнозирование дивидендной доходности ценных бумаг:

- «Dividend.com»;
- «Robinhood»;
- «Finbox».

Рассмотрим каждый программный продукт более подробно.

1.3.1 Программное средство «Dividend.com»

Dividend.com - это онлайн-платформа, которая предоставляет инвесторам информацию о дивидендах и помогает им принимать более обоснованные инвестиционные решения на основе этой информации. Сайт содержит базу данных с информацией о более чем 4 000 акциях, которые выплачивают дивиденды. Кроме того, на платформе можно найти информацию о календаре дивидендов, исторических данных, новостях и аналитических обзорах.

С помощью Dividend.com инвесторы могут легко отслеживать выплаты дивидендов своих инвестиций и получать уведомления об изменениях в выплатах. Также платформа предоставляет инструменты для анализа и сравнения дивидендной доходности различных акций, что помогает инвесторам принимать более обоснованные инвестиционные решения.

Платформа предоставляет услуги как для индивидуальных инвесторов, так и для финансовых профессионалов. Платная подписка на сайт предоставляет пользователям расширенный доступ к инструментам анализа, прогнозирования и портфельному управлению.

Пример функционала программного средства представлен на рисунке 1.2.

Преимущества программного средства:

- комплексная информация. Программное средство предоставляет обширную базу данных о дивидендах и доходности инвестиций на финансовых рынках. Здесь можно найти данные о более чем 18 тысячах акций, более 500 фондов и многих других инвестиционных возможностях;
- удобство использования. Платформа имеет простой и удобный интерфейс, который позволяет пользователям легко находить нужную информацию, проводить анализы и принимать инвестиционные решения;
- объективность. Все данные на платформе Dividend.com основаны на фактах, что обеспечивает объективность и надежность информации;
- инструменты анализа. Портал предлагает различные инструменты анализа, которые позволяют пользователям проводить исследования и оценку инвестиционных возможностей на финансовых рынках;
- новости и статьи. Платформа предоставляет своим пользователям доступ к последним новостям и статьям из мира инвестиций, что позволяет

быть в курсе последних тенденций на финансовых рынках и принимать обоснованные решения;

- персонализация. Dividend.com позволяет пользователям настраивать уведомления о новостях и событиях на рынке в соответствии с их инвестиционными интересами;

- сообщество. Платформа создает сообщество инвесторов, где пользователи могут общаться, обмениваться мнениями и опытом, что также способствует повышению качества инвестиционных решений.

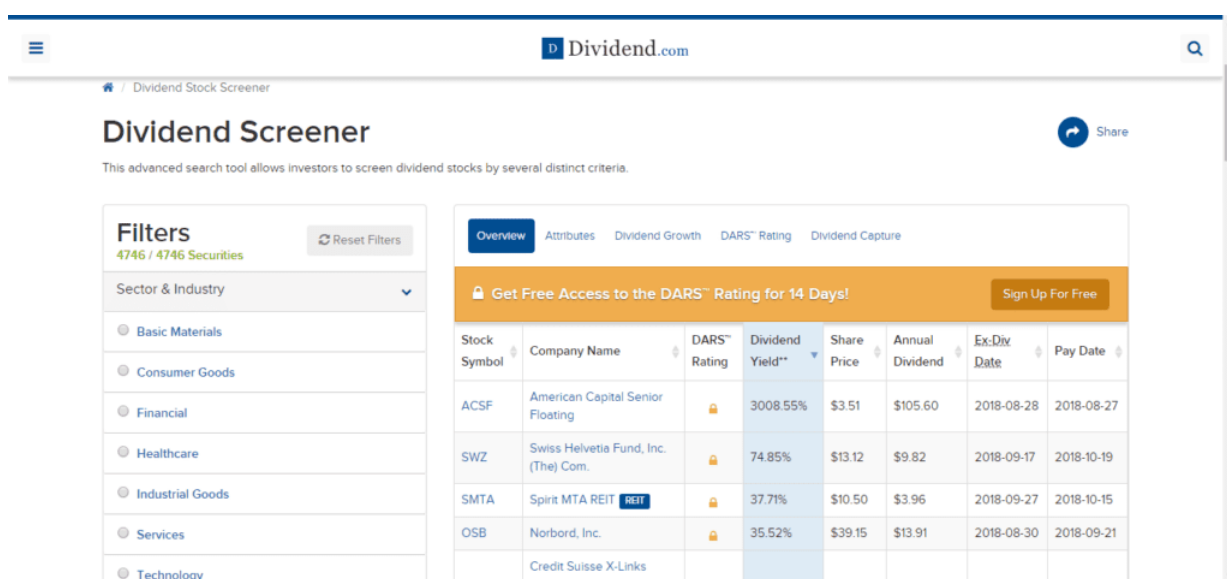


Рисунок 1.2 – Интерфейс окон программного обеспечения «Dividend.com»

1.3.2 Программное средство «Robinhood»

Robinhood - это финтех-платформа, которая позволяет инвесторам торговать на финансовых рынках США. Она была основана в 2013 году в Калифорнии и с тех пор стала одной из самых популярных инвестиционных платформ в США.

Основной функционал включает в себя бесплатную покупку и продажу акций, фондов, опционов и криптовалюты. Кроме того, пользователи могут просматривать актуальную информацию о ценах на акции, новости и аналитические материалы, а также настраивать уведомления и другие параметры своих инвестиционных операций.

Одним из главных преимуществ программного средства является его бесплатность: платформа не берет комиссию за торговые операции с акциями и фондами. Более того, Robinhood также предоставляет пользователям бесплатные опционные сделки и торговлю криптовалютой.

Портал также предлагает своим пользователям удобный мобильный интерфейс, который позволяет быстро и просто управлять своим портфелем инвестиций, делать покупки и продажи, а также получать актуальную информацию о рынке в режиме реального времени.

Однако, следует отметить, что Robinhood не является лицензированным инвестиционным консультантом, и не предоставляет индивидуальные финансовые советы своим пользователям. Кроме того, на платформе не предусмотрена возможность торговли на других финансовых рынках, кроме США.

Графический интерфейс программы представлен на рисунке 1.3.

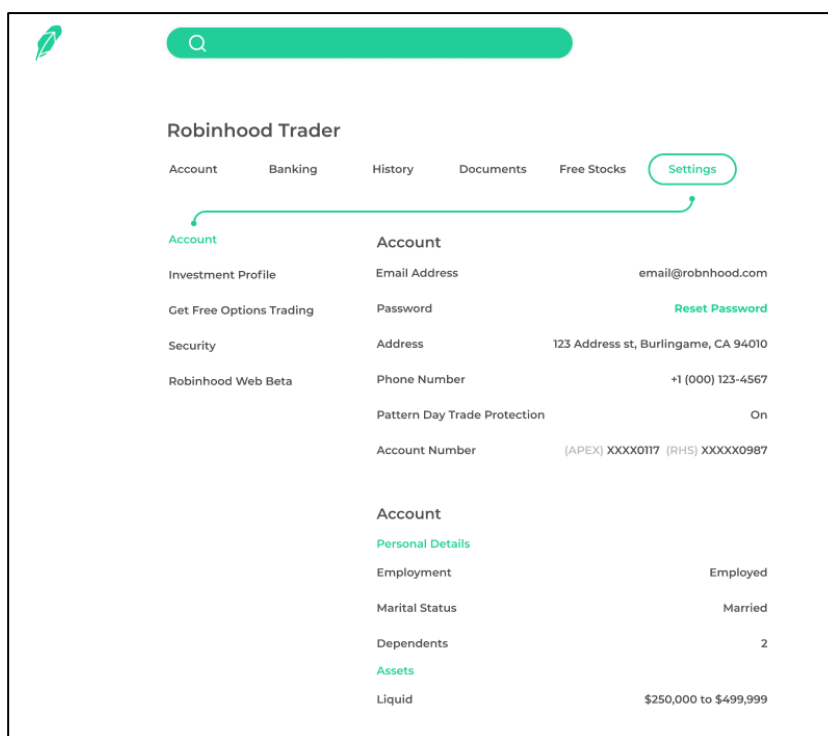


Рисунок 1.3 – Интерфейс программного средства «Robinhood»

Преимущества Robinhood:

- программное средство не берет комиссию за торговые операции с акциями, фондами и опционами, что делает торговлю на платформе более доступной для малых инвесторов;
- приложение имеет простой и удобный интерфейс, который позволяет быстро и просто совершать торговые операции;
- на портале нет минимального требования к депозиту, что также делает платформу доступной для малых инвесторов;
- на сервисе можно торговать не только акциями, но и фондами, опционами и криптовалютой, что расширяет возможности инвестирования;
- портал предоставляют обучающие материалы, что помогает новичкам разобраться в инвестиционных стратегиях.

Недостатки Robinhood:

- на портале можно торговать только на американском рынке, что ограничивает возможности инвестирования;
- сервис не предоставляет индивидуальных финансовых советов, что может быть недостатком для инвесторов, которые нуждаются в персональном подходе;
- программное средство предлагает ограниченный выбор инструментов для торговли, поэтому не все инвесторы могут найти на платформе интересующие их акции и фонды;
- как и любая другая технологическая платформа, сервис может столкнуться с техническими проблемами, такими как задержки при обработке транзакций или сбои в работе приложения;
- любая торговля на финансовых рынках связана с риском потерь, и пользователи не застрахованы от потерь своих инвестиций.

1.3.3 Программное средство «Finbox»

Finbox Screener - это онлайн-сервис, который предоставляет инвесторам возможность фильтровать акции по определенным критериям и находить наиболее подходящие для инвестирования.

Основные возможности Finbox Screener включают в себя:

- фильтрация акций по множеству критериев, таким как рыночная капитализация, доходность дивидендов;
- создание пользовательских фильтров для анализа акций;

- отображение ключевых показателей и финансовых показателей для каждой отфильтрованной акции;
- графическое представление основных финансовых показателей и их динамики на протяжении нескольких лет;
- сравнение акций в разных секторах и отраслях.

На рисунке 1.4 представлен интерфейс программы:

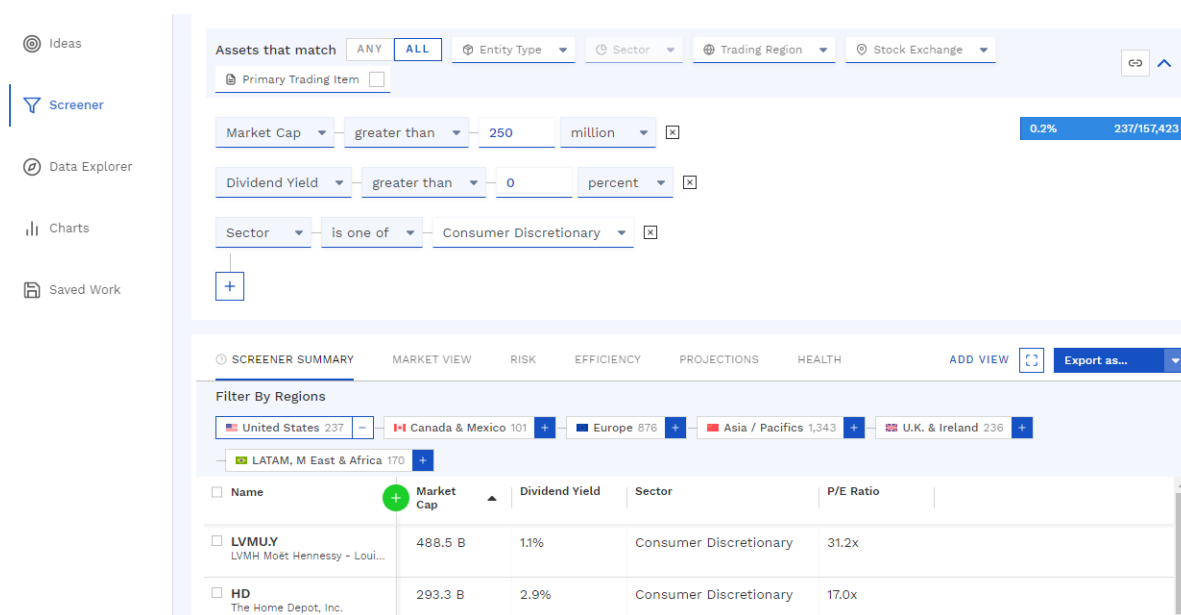


Рисунок 1.4 – Интерфейс программного средства «Finbox»

Существует множество сервисов для анализа финансовых данных и инвестиционных возможностей, и Finbox Screener имеет несколько отличий от них:

- предоставляет широкий набор фильтров для анализа акций, включая пользовательские фильтры, что позволяет инвесторам проводить качественный анализ и выбирать акции, соответствующие их инвестиционным стратегиям.
- сервис предоставляет графическое представление основных финансовых показателей и их динамики на протяжении нескольких лет, что позволяет быстро оценить их изменение.
- ориентирован на инвесторов, что делает его более удобным для использования, чем многие другие сервисы;

- сервис имеет интуитивно понятный интерфейс и легок в использовании, что делает его доступным для широкого круга пользователей.

- предоставляет бесплатную версию сервиса, которая включает в себя базовый набор функций для анализа акций.

Однако, как и любой другой сервис, Finbox Screener не является идеальным и имеет свои недостатки, такие как ограничения в использовании бесплатной версии и недостаточная точность финансовой информации для некоторых акций.

Таким образом, были рассмотрены наиболее востребованные программные средства на рынке. Были рассмотрены цели и задачи данных систем, их базовый и дополнительный функционал, а также интерфейс пользователя. Все это будет учитываться при дальнейшей постановке задачи на разработку программного средства.

1.4 Разработка информационной модели предметной области

При анализе предметной области и разработке функциональной спецификации разрабатываемого проекта были выделены следующие 5 сущностей.

«Пользователь» - сущность, представляющего пользователя программного средства. Рассматриваемая сущность представлена следующими полями:

- код – уникальный идентификатор пользователя;
- логин – логин пользователя;
- пароль – хэш-код пароля пользователя;
- код роли – идентификатор роли пользователя.

«Роль» - сущность, представляющая разновидность учетной записи, которая выдается администратором портала в зависимости от необходимого функционала в программном средстве. Рассматриваемая сущность представлена следующими полями:

- код – уникальный идентификатор роли;
- наименование – наименование роли;

«Ценная бумага» - сущность, представляющая собой ценную бумагу, которая представлена в программном средстве для расчета ее дивидендной доходности. Сущность представлена следующими полями:

- код – уникальный идентификатор бумаги;
- компания – наименование компании-собственника;
- характеристика обращаемости;
- форма существования;
- срок существования;
- дата выпуска;
- стоимость единицы;
- дивиденды.

«Запись изменения стоимости» - сущность, представляющая собой запись, свидетельствующая об изменении стоимости ценной бумаги и предназначена для ведения истории изменений ее стоимости. Сущность представлена следующими полями:

- код – уникальный идентификатор записи;
- дата – дата изменения стоимости;
- сумма – сумма, которая сменила предыдущую за единицу;
- код ценной бумаги – идентификатор ценной бумаги, которой соответствует запись.

«Лог» - сущность, представляющая собой запись о действиях пользователя в системе. Сущность представлена следующими полями:

- код – уникальный идентификатор записи;
- код пользователя - уникальный идентификатор пользователя, совершившего действие;
- сообщение – сообщение о действии пользователя;
- дата – дата, когда было совершено действие.

Схема информационной модели программного средства представлена на рисунке 1.5:

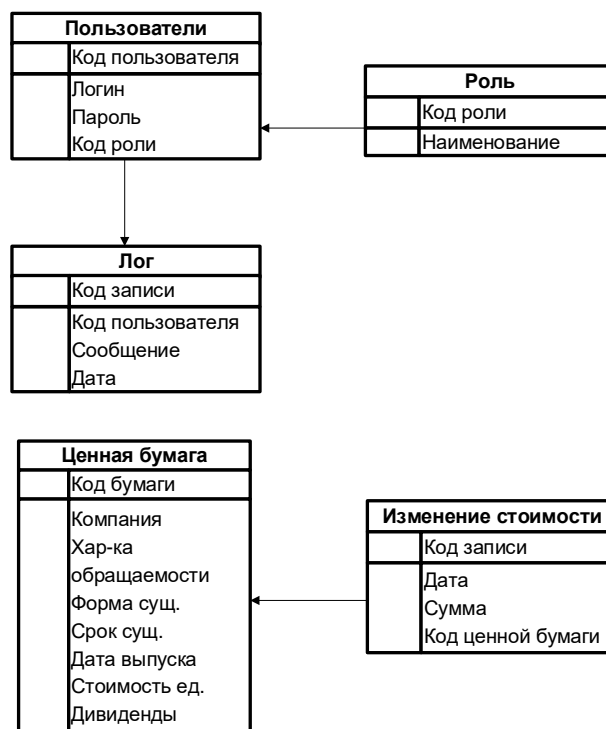


Рисунок 1.5 – Информационная модель программного средства

1.5 UML-модели представления программного средства и их описание

UML (Unified Modeling Language) - это стандартный язык моделирования, который используется для описания, проектирования и документирования программных систем. Он позволяет разработчикам визуализировать различные аспекты системы и увидеть, как они связаны между собой. [2]

Диаграммы UML имеют множество преимуществ при разработке ПО:

- они позволяют представить сложные аспекты программной системы в виде графических диаграмм, что облегчает понимание для всех участников проекта, включая менеджеров, аналитиков и программистов;
- с помощью диаграмм UML можно выявить потенциальные проблемы и ошибки в системе на ранних стадиях проекта, что позволяет сократить время на разработку и исправление ошибок;

- диаграммы UML помогают улучшить коммуникацию между различными участниками проекта и облегчают передачу информации;
- использование диаграмм UML позволяет разработчикам легко вносить изменения в программную систему на любой стадии проекта, что повышает гибкость разработки;

Однако, необходимо отметить, что UML может иметь и некоторые недостатки, например:

- диаграммы UML могут быть сложными и трудными для понимания для неопытных разработчиков или пользователей;
- UML может привести к созданию слишком многих диаграмм, что может затруднить процесс разработки;
- использование UML может привести к некоторым неоднозначностям в интерпретации диаграмм разными участниками проекта;

Несмотря на эти недостатки, диаграммы UML остаются важным инструментом для разработки программного обеспечения и позволяют значительно улучшить качество и эффективность разработки.

Диаграмма состояний системы для варианта использования «Изменить пользователя» приведена на рисунке 1.6.

В ходе варианта использования «Изменить пользователя» система принимает следующие состояния:

- вывод формы изменения данных (наступает при начале варианта использования, а также в случаях, когда не пройдена валидация или неудачно сохранение записи в базе данных);
- валидация данных формы (наступает при передаче формы на сервер). Валидация предусматривает собой проверку целостности данных, а также удовлетворение условиям сервера;
- сохранение записи в базе данных (наступает при успешном сохранении файла).

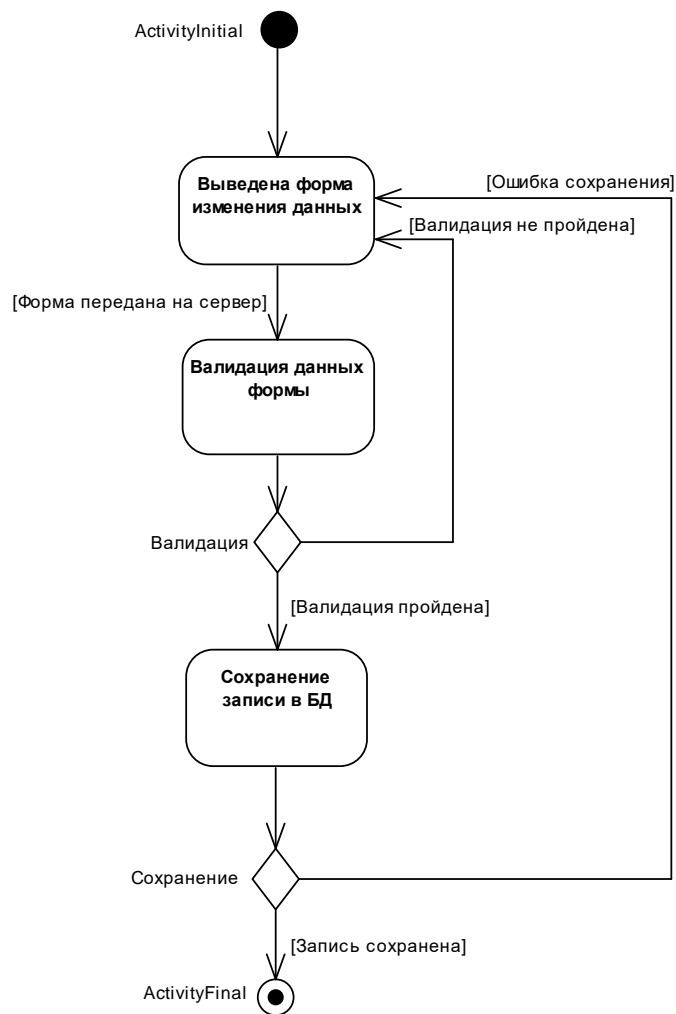


Рисунок 1.6 – Диаграмма состояний «Изменить пользователя»

Диаграмма развертывания системы приведена на рисунке 1.7:

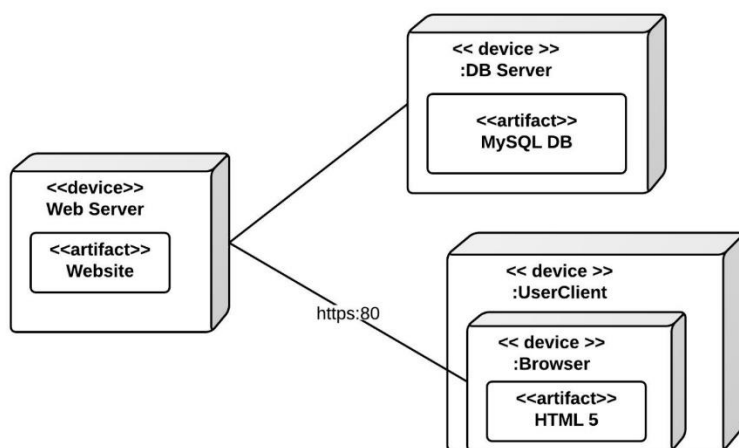


Рисунок 1.7 – Диаграмма развертывания системы

Диаграмма развертывания - это одна из диаграмм UML, которая отображает аппаратное и программное обеспечение, которые необходимы для развертывания системы, а также их связи и конфигурацию. Данная диаграмма используется для визуализации структуры системы на стадии развертывания, показывая, как система будет запущена на конкретном оборудовании. [8]

Диаграмма развертывания для программного средства состоит из трех узлов, а именно:

- сервер баз данных;
- веб-сервер;
- клиент.

Сервер баз данных представляет собой программно-аппаратный комплекс, задачей которого является надежное и стабильное функционирование базы данных.

Веб-сервер служит для аккумуляции запросов от клиента, обращений в базу данных, а также выполнения объемных и затратных арифметических операций.

Клиент служит для отображения пользовательского интерфейса и предоставления функционала для взаимодействия с сервером. Клиент также генерирует запросы к серверу.

2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

2.1 Постановка задачи

При сравнении программ конкурентов были выявлены следующие параметры, которые должна содержать разрабатываемая информационная система: понятный и удобный в использовании интерфейс программы; хранение и отображение данных о ценных бумагах; предоставление аналитической информации; недопущение несанкционированного доступа в систему.

Исходя из данных параметров можно увидеть, что сравненные нами программы не отвечают всем пользовательским требованиям, таким как:

Проведение операций над данными предприятия требует значительного количества времени. Это обусловлено запутанным и сложным интерфейсом, что не очень удобно для пользователей.

Для использования программ конкурентов необходимо приобретать их лицензии или обновления, что требует большого количества денежных затрат. Кроме того, некоторые программные продукты имеют настолько сложный интерфейс, что инвесторам необходимо покупать курсы по использованию данных программных средств.

Некоторые малые и средние инвесторы, которые не в состоянии приобрести в пользование лицензии дорогостоящих программ, ведут учет данных в бумажном виде, что негативно сказывается на скорости и качестве анализа рынка перед принятием инвестиционного решения.

Учитывая все вышеперечисленные основания было решено разработать собственную информационную систему, которая была бы удобна и практична в использовании.

Название программного продукта: «ПРОГНОЗ».

Курсовой проект будет разработан с помощью языка программирования JavaScript. Графический интерфейс реализуется с помощью языка разметки HTML, а также с помощью языка декорирования CSS. Хранение данных организовано в СУБД MySQL Server.

Целью создания данного программного продукта будет оптимизация прогнозирования дивидендной доходности ценных бумаг.

Программа должна иметь следующие параметры и функциональные возможности:

- содержать базу данных, необходимую для хранения, изменения и удаления данных;
- возможность добавлять, удалять и изменять ценными бумагами;
- возможность управлять пользователями системы;
- предоставлять графический интерфейс для расчета доходности ценных бумаг;
- выводить аналитический график;

Кроме перечисленных возможностей, информационная система должна содержать понятный интерфейс, а также не иметь необходимости в финансовых вложениях.

2.2 Обоснование выбора компонентов и технологий для реализации программного средства

Правильный выбор средств разработки на этапе проектирования является ключевым моментом в процессе разработки программного обеспечения. Это важно по многим причинам. Правильный выбор средств разработки может значительно повысить эффективность процесса разработки, уменьшить время, необходимое для создания программного продукта, а также снизить затраты на разработку. Также, правильный выбор средств разработки может повысить качество программного обеспечения. Некоторые инструменты могут быть более подходящими для определенных задач, что может повысить надежность и безопасность продукта. Правильный выбор средств разработки может облегчить сопровождение и поддержку программного обеспечения в будущем. Некоторые инструменты и технологии могут быть более подходящими для дальнейшего развития и сопровождения программного продукта. [8]

Кроме того, правильный выбор средств разработки может помочь разработчикам эффективнее использовать свои навыки и опыт, что может привести к более удачному решению задач и более качественному результату.

Среди технологий, применяемых при разработке веб-приложений, наиболее распространенными в настоящее время являются следующие:

1. Node.js - это кросс-платформенная среда выполнения JavaScript с открытым исходным кодом. [1]

2. ASP.NET - это фреймворк, разработанный компанией Microsoft для создания веб-приложений на платформе .NET. Он включает в себя средства для разработки, тестирования и развертывания веб-приложений, а также интегрируется с другими продуктами Microsoft, такими как Visual Studio и SQL Server.

3. AMP - это стек технологий, включающий в себя Apache, MySQL и PHP. Он используется для создания веб-приложений на основе сервера Apache, базы данных MySQL и языка программирования PHP. AMP является открытым и бесплатным набором технологий, доступным для всех пользователей.

Сравнение приведенных платформ приведено в таблице 2.1.

Таблица 2.1 – Сравнение платформ разработки веб-приложений

Критерий	Node.js	.NET Framework	AMP
Кроссплатформенность	Да	Нет	Да
Локализованная документация	Частично	Да	Да
Веб-сервер	Express, Koa, Napi	IIS, Apache	Apache
Сервер приложений	Java EE	Фреймворк .NET	Среда исполнения PHP
СУБД	Oracle, PostgreSQL, MySQL	MS SQL, MySQL, Oracle, Sybase	MySQL
Язык программирования	JavaScript	C#, VB, C++	PHP

На основании таблицы 1.6 в качестве платформы реализации выберем Node.js по причине легкости в развертывании приложения и отсутствия жестких правил при построении архитектуры программного средства.

Для настоящего проекта выбрана технология Node.js по причине большей гибкости в реализации серверной составляющей [1].

При формировании интерфейса пользователя, выводимого в веб-браузер в виде веб-страниц, предполагается использование следующих фреймворков и библиотек:

- bootstrap 4 для адаптивной верстки веб-интерфейса [3];
- css – язык декорирования [4].

При реализации приложений на базе Node.js наиболее часто применяются следующие СУБД: Microsoft SQL, MySQL, PostgreSQL.

Сравнение приведенных СУБД приведено в таблице 2.2.

Таблица 2.2 – Сравнительный анализ СУБД

Критерий	SQLite	MySQL	PostgreSQL
Кроссплатформенность (Windows, Linux)	+ / –	+ / +	+ / +
Ограничение на использование RAM, GB	не ограничено	не ограничено	не ограничено
Ограничение на использование CPU, количество ядер	не ограничено	не ограничено	не ограничено
Лицензия	GPL	GPL	GPL
Скорость работы	Средняя	Высокая	Средняя

Таким образом, в сравнении с конкурентами СУБД MySQL Server имеет преимущество, так как имеет более высокую скорость работы.

2.3 Архитектурные решения

С учетом наличия в функциональной схеме программного средства базы данных, было принято решение применить в качестве архитектурного решения программного продукта архитектуру клиент-сервер.

Составными частями данной архитектуры являются клиент и сервер. Клиент – программное обеспечение на стороне пользователя. Его роль заключается в том, чтобы отправлять запросы на сервер для доступа к данным либо для выполнения каких-либо операций, требующих вычислительных мощностей. Сервер – более мощная с точки зрения

аппаратных ресурсов система, которая оперирует размещенными на ней данными, предоставляя клиенту доступ либо данные по его запросу.

Клиент и сервер взаимодействуют друг с другом с помощью запросов, которые отправляет клиент, на которые сервер, проводя какие-либо операции с данными, отправляет ответ. В случае, если приходит несколько запросов, сервер отвечает на них последовательно, формируя очередь. В некоторых системах запросы клиента имеют приоритет, в зависимости от которого сервер выстраивает очередь. На рисунке 3.3 представлена схема построения архитектуры «клиент-сервер». [5]

Основными достоинствами архитектуры «клиент-сервер» являются:

- возможность, в большинстве случаев, распределить функциональность системы между несколькими вычислительными узлами в сети, позволяя упростить обслуживание вычислительной системы;
- все данные хранятся на выделенном сервере, к которому, как правило, предъявляются повышенные требования к защите, защищая данные от несанкционированного доступа;
- позволяет работать с несколькими клиентами одновременно;
- использовать ресурсы серверной части системы чаще всего могут клиенты с различными аппаратными платформами, операционными системами и так далее.

Однако архитектура «клиент-сервер» не лишена и недостатков:

- необходимость обеспечения дорогостоящими аппаратными ресурсами;
- при высокой степени интеграции или при использовании централизованной системы, в случае отказа одного из компонентов (в случае централизованной системы это основной сервер) всё приложение также станет неработоспособным;
- система требует своевременного квалифицированного технического обслуживания. [6]

Таким образом, учитывая специфику работы программного средства и учитывая все особенности разрабатываемого проекта было принято решение использовать архитектуру клиент-сервер. Это позволит разграничить данные и пользовательский интерфейс.

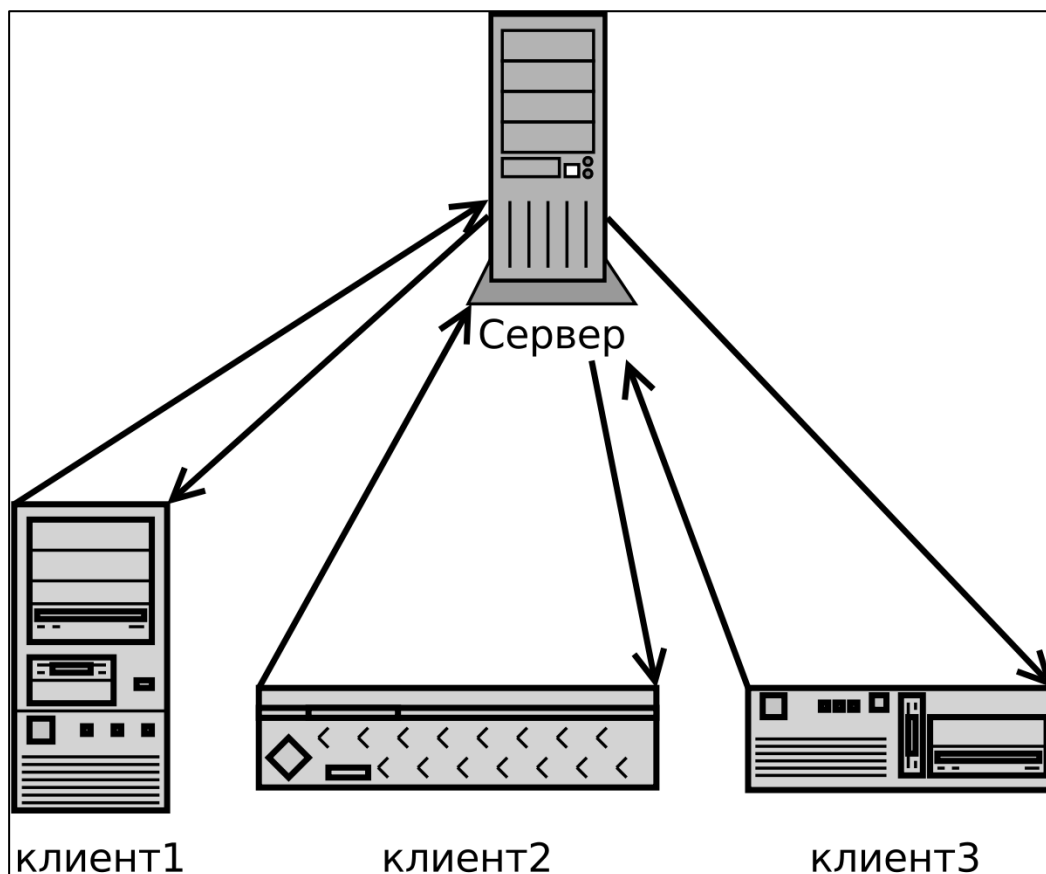


Рисунок 2.1 – Двухуровневая архитектура «клиент-сервер»

2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства

Одним из основных и первых при разработке программы алгоритмов – алгоритм добавления ценных бумаг, схема программы для данного алгоритма представлена на рисунке 2.2.

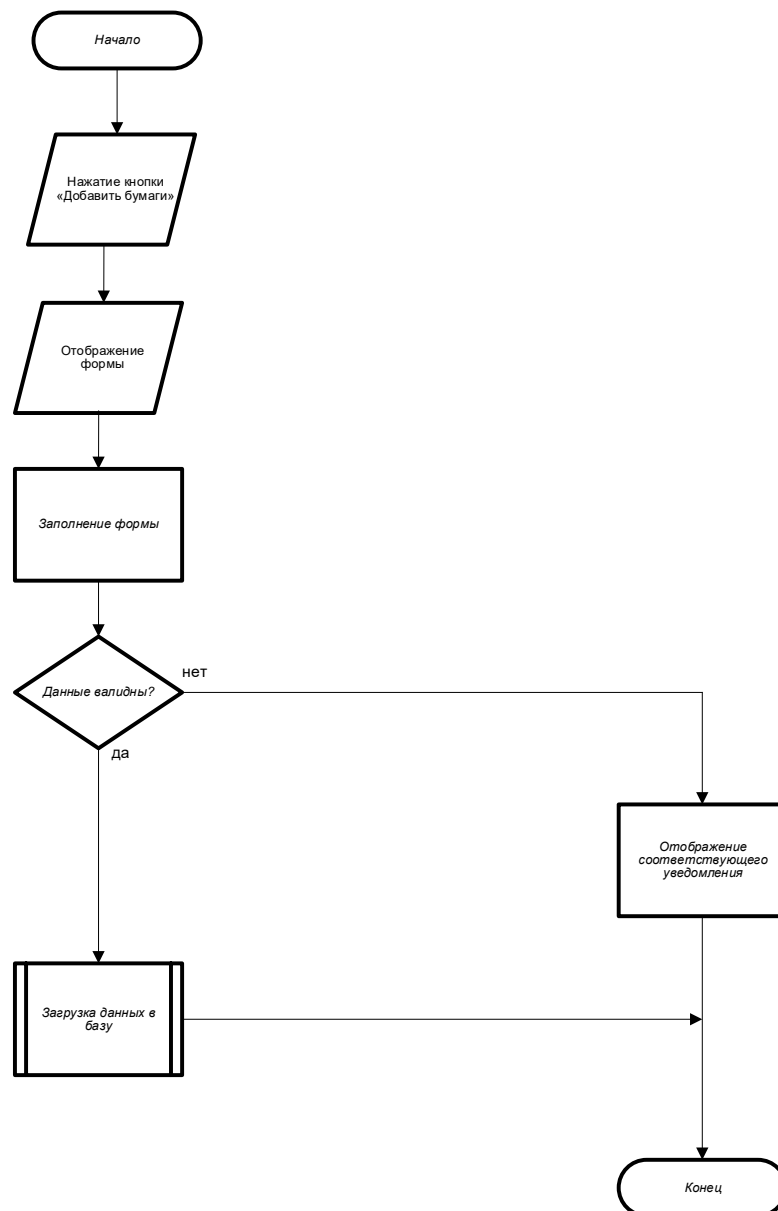


Рисунок 2.2 – Схема программы для алгоритма добавления ценных бумаг

Приведенный алгоритм реализуется методами `addPaper` и предполагает следующий порядок действий:

- нажатие кнопки «Добавить бумаги»;
- отображение формы для ввода необходимой информации о ценных бумагах;
- заполнение формы пользователем;
- проверка данных на валидность на стороне сервера;
- если данных валидны, то происходит добавление их в базу данных, а если нет – вывод сообщения с соответствующим уведомлением;

Далее, рассмотрим калькуляции дивидендной доходности ценных бумаг, схема программы для данного алгоритма представлена на рисунке 2.3:

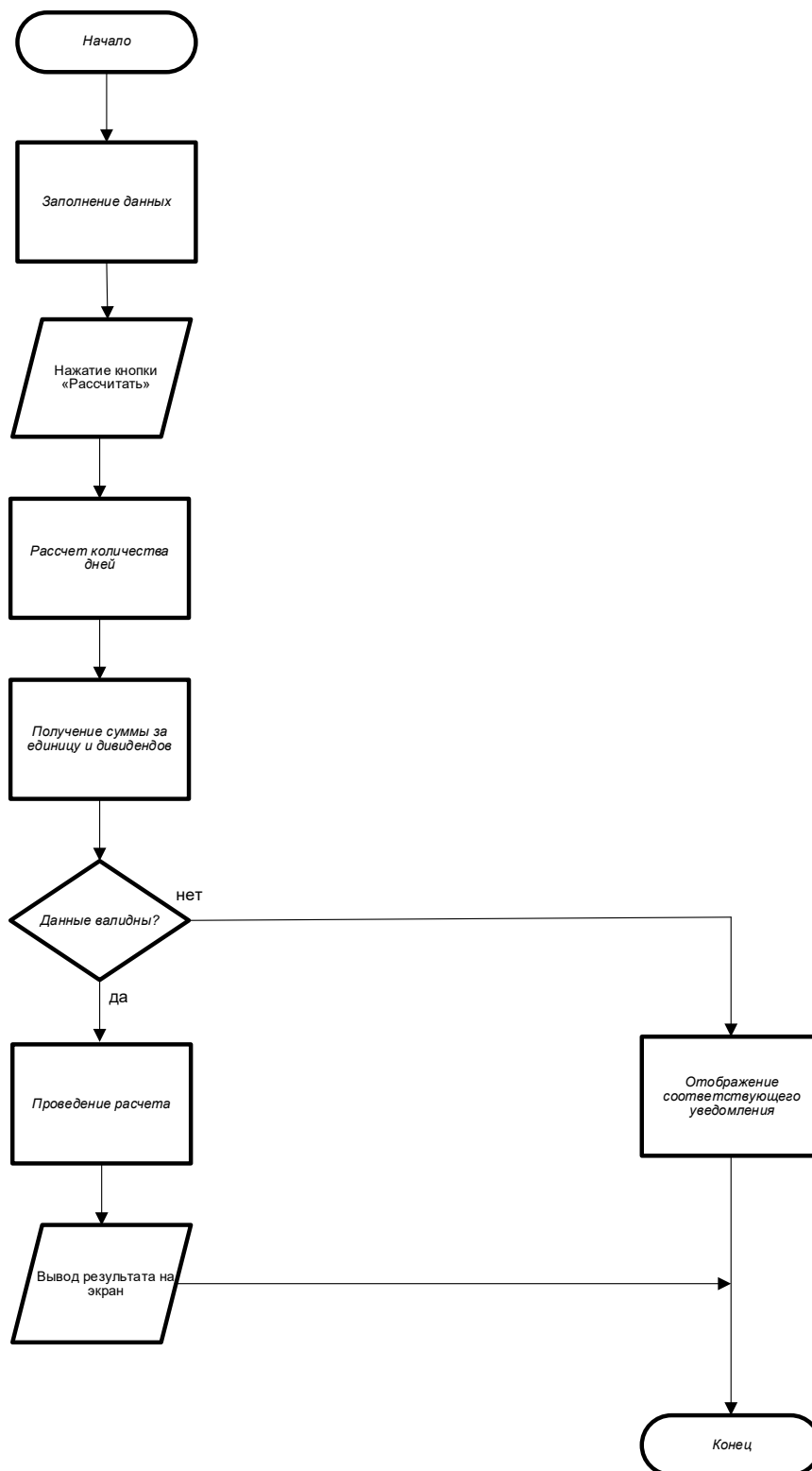


Рисунок 2.3 – Схема программы для расчета доходности ценных бумаг

Приведенный алгоритм реализуется методами `sum()` класса и предполагает следующий порядок действий:

- заполнение данных о количестве бумаг и дате получения прибыли, а также выборе бумаги из списка;
- нажатие кнопки «Рассчитать»;
- расчет количества дней между сегодняшним днем и днем получения прибыли;
- получение суммы за единицу и процентов дивидендов из соответствующих полей формы;
- проверка данных на валидность;
- проведение расчета;
- вывод результата на экран.

2.5 Проектирование пользовательского интерфейса

Интерфейс (от англ. *interface*) - в широком смысле это определенная стандартами граница между взаимодействующими независимыми объектами. [2]

Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов. Он определяет:

1. язык пользователя;
2. язык сообщений компьютера, организующий диалог на экране дисплея;
3. знания пользователя.

Язык пользователя — это действия, производимые пользователем при работе с системой путем использования возможностей клавиатуры, пишущих на экране электронных устройств, джойстика, мыши, подаваемых голосом команд и т. п. Наиболее простой формой языка пользователя является создание форм входных и выходных документов. Получив входную форму (документ), пользователь заполняет его необходимыми данными и вводит в компьютер. В результате проведения компьютерной программой определенных автоматизированных процедур можно получать различные результаты, например, в виде выходного документа установленной формы.[2]

Язык сообщений — это информация, видимая пользователем на экране дисплея (символы, графика, цвет); это также полученные на принтере данные, звуковые выходные сигналы и т. п.

Важным измерителем эффективности используемого интерфейса является выбранная форма диалога между пользователем и системой.

Диалог (человеко-машинный диалог) представляет собой последовательность запросов пользователя, ответов на них компьютера и наоборот (запрос пользователя, ответ и запрос компьютера, окончательное действие компьютера и др.). Он осуществляется в процессе выполнения каких-либо действий путем взаимодействия пользователя с компьютером.

Наиболее распространены такие формы диалога: запросно-ответный режим, командный режим, режим меню, режим заполнения предлагаемых компьютером пропусков в выражениях. [2]

Каждая форма в зависимости от типа задачи, особенностей пользователя и принимаемого решения имеет достоинства и недостатки. Долгое время единственной реализацией языка сообщений был отпечатанный или выведенный на экран дисплея отчет или сообщение.

В ЭВМ представление выходных данных осуществляется с помощью машинной (компьютерной) графики. Она позволяет создавать на экране и бумаге цветные графические изображения в двумерном и трехмерном виде. Использование такой графики значительно повышает наглядность и интерпретацию выходных данных, все чаще используется в ИТ поддержки принятия решений.

Пользовательский интерфейс (интерфейс пользователя) (от англ. user interface) в информационных технологиях — это элементы и компоненты программы, оказывающие влияние на взаимодействие пользователя с программным обеспечением; это совокупность правил, методов и программно-аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером.

Графический интерфейс пользователя — это графическая среда организации взаимодействия пользователя с вычислительной системой, предполагающая стандартное использование основных элементов диалога пользователя с ЭВМ.

К графическим интерфейсам относят все оконные чисто графические системы — это Windows, оболочки для UNIX — X-Window, Photon из ОС

QNX, Aqua из MacOS X. Основное преимущество его использования в операционной системе (далее — ОС) заключается в том, что он позволяет создавать одинаковые графические изображения для всех устройств, поддерживаемых ОС, реализуя принцип WYSIWYG (от англ. What You See Is What You Get — что видим, то и получаем). [8]

Графический интерфейс позволяет управлять поведением вычислительной системы через визуальные элементы управления: окна, списки, кнопки, гиперссылки и полосы прокрутки. Он включает такие понятия, как: рабочий стол, окна, пиктограммы, элементы графического интерфейса, манипуляция указывающим устройством (мышь). Эти визуальные элементы создаются, отображаются и обрабатываются с помощью графических приложений.

Для визуализации рабочих плоскостей будущего интерфейса, а также для оценки качества проектирования пользовательского интерфейса с точки зрения удобства пользования, была разработана карта веб-страниц программного средства (рисунок 2.4):

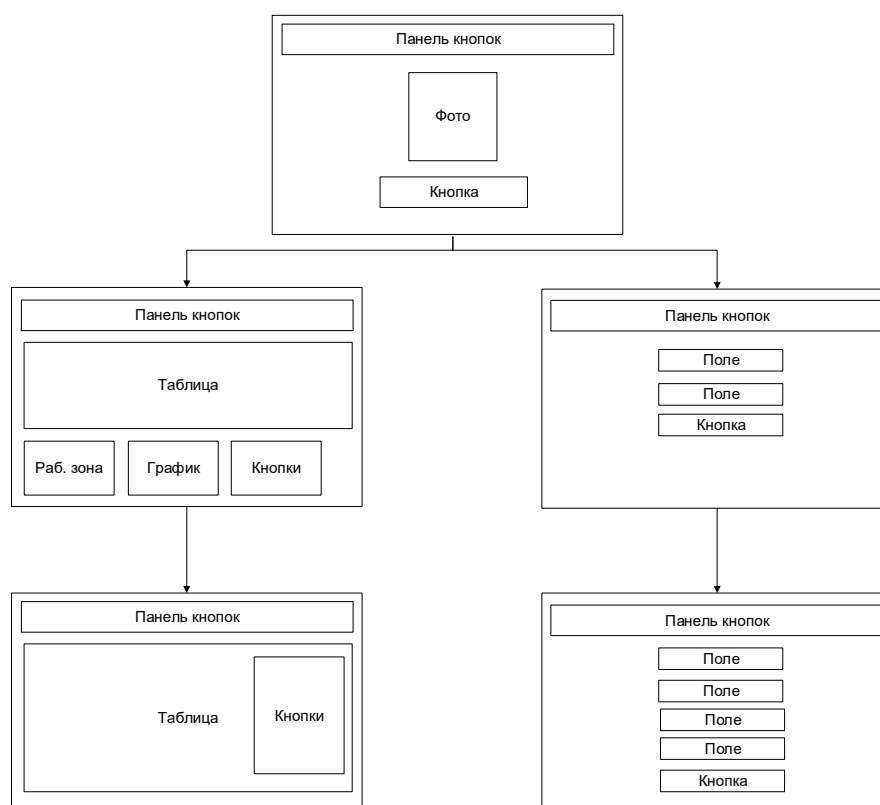


Рисунок 2.4 – Карта веб-страниц программного средства.

2.6 Методы и средства, используемые для обеспечения безопасности данных

Самая распространенная защита компьютерной информации - защита на основе пароля. При реализации парольной защиты вход в систему, запуск приложения, запрос на доступ к данным сопровождается запросом пароля и последующим сравнением введенного пароля с оригиналом.

Пароль представляет собой последовательность символов некоторого алфавита и специальных знаков. Последовательность должна удовлетворять ограничению на наименьшую и наибольшую длину.

Стоит обратить внимание, что парольная защита может быть рекомендована только для использования при защите информации, предназначенной для узкого круга пользователей. При широком использовании, например, программ, защищенных таким образом, очень велика вероятность того, что, хотя бы один законный пользователь сообщит пароль злоумышленнику, этого будет достаточно для того, чтобы сделать защищенное приложение общедоступным.

Следует обратить внимание на то, что процесс ввода пароля поддается наблюдению, даже в том случае, если отсутствует режим «эхо». Человек, находящийся рядом с пользователем, вводящим пароль, наблюдая за процессом набора на клавиатуре, может зафиксировать вводимые символы. Кроме того, существует множество специальных программ типа «тройанский конь», которые через перехват соответствующего прерывания читают и сохраняют пароли, набираемые на клавиатуре.

Оригинальный пароль необходимо хранить в месте, доступном защищенному приложению и малодоступном злоумышленнику.

На практике для хранения эталонного пароля используются следующие способы:

- 1) эталон хранится непосредственно в защищенной программе;
- 2) эталон хранится в отдельном специально предназначенном файле;
- 3) эталон хранится в системных областях (в свободных, зарезервированных или редко используемых областях дисков, системных базах данных). Например, при защите Windows-приложений разработчики часто хранят оригинальный пароль в системной базе данных Registry (системный реестр).

Так как, например, в случае хранения пароля непосредственно в защищаемой программе злоумышленник может легко найти эталонный пароль, либо просмотрев дампы файла, в котором хранится программа, либо даже с помощью специальной программы, распечатывающей все текстовые строки.

В основном авторы защит либо шифруют пароль известными или собственными криптографическими методами, либо применяют хэш-функции (хэш-коды) для преобразования пароля.

Метод хэширования пароля заключается в хранении в качестве эталона не собственно пароля, а результата определенных автором защиты математических преобразований (именно эти преобразования и называются хэш-функцией или хэшированием) над символами пароля. При запуске приложения введенный пользователем пароль подвергается хэшированию и сравнению полученного результата с эталоном. [6]

3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Для тестирования разработанного программного обеспечения использовано ручное тестирование по методу дымовых тестов.

Дымовое тестирование рассматривается как короткий цикл тестов, выполняемый для подтверждения того, что после сборки кода (нового или исправленного) устанавливаемое приложение, стартует и выполняет основные функции. Вывод о работоспособности основных функций делается на основании результатов поверхностного тестирования наиболее важных модулей приложения на предмет возможности выполнения требуемых задач и наличия быстронаходимых критических и блокирующих дефектов. В случае отсутствия таковых дефектов дымовое тестирование объявляется пройденным, и приложение передается для проведения полного цикла тестирования, в противном случае, дымовое тестирование объявляется проваленным, и приложение уходит на доработку.[8]

Проверка выполнения требований к функциям системы проводилась на основании подготовленных тест-кейсов. При таком тестировании требуется выполнить ряд действий, при этом результат выполнения тестируемой функции программного обеспечения должен быть заранее известен. [9]

Результаты тестирования приведены в таблице 3.1.

Таблица 3.1 – Результаты тестирования

Вариант использования / Модуль	Описание теста	Ожидаемый результат	Статус теста
1	2	3	4
Авторизация пользователя в системе / Login	Вводит логин и пароль пользователя	Проверяет данные пользователя	пройден
Регистрация в системе / Reg	Вводит необходимые данные в форму	Сохраняет данные в базу, предлагает авторизоваться	пройден
Изменить пользователя / modifyUser	Выбирает пользователя, редактирует данные	Сохраняет и отображает отредактированные данные	пройден

Продолжение таблицы 3.1

1	2	3	4
Просмотр аналитического графика / getGraf	Выбор ценной бумаги в таблице	Отображает график изменения цены	пройден
Удалить пользователя / deleteUser	Выбирает пользователя, нажимает кнопку «Удалить»	Пользователь удаляется, интерфейс обновлен	пройден
Сбросить пароль пользователя / resetPassword	Выбирает пользователя, нажимает кнопку «Сбросить пароль»	Сброс пароля пользователя до стандартного	пройден
Попытка авторизации с неверными данными / Login	Умышленно вводит неверные данные авторизации	Доступ к приложению отсутствует, программа выдает соответствующее уведомление	пройден
Регистрация с несовпадающими паролями / Reg	Умышленно вводит несовпадающие пароли в форме	Добавление данных в базу не происходит, программа отображает соответствующее уведомление	пройден
Удалить ценную бумагу / DeleteStaff	Выбирает ценную бумагу в списке и нажимает кнопку «Удалить»	Удаление данных о сотруднике из базы данных	пройден

Прочие функции протестированы с использованием аналогичных тестов. Все тесты пройдены.

4 РУКОВОДСТВО ПО УСТАНОВКЕ (РАЗВЕРТЫВАНИЮ) И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

4.1 Руководство по установке (развертыванию) программного средства

Рассмотрим развертывание приложения Node.js на примере развертывания приложения на одном из самых востребованных хостингов – hoster.by.

В панели управления хостингом (cPanel) необходимо выбрать в разделе «Программное обеспечение» пункт «Настройка Node.js приложений» (рисунок 4.1):

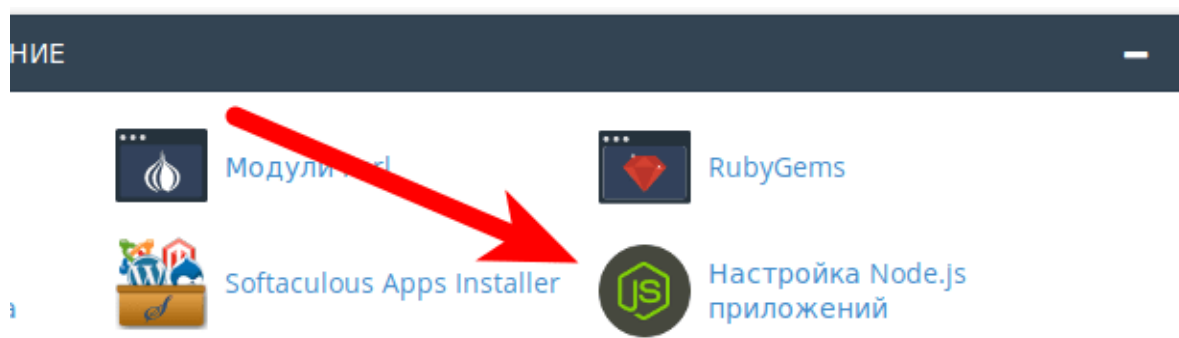


Рисунок 4.1 – Выбор типа проекта

Далее, необходимо выбрать в появившемся окне «Создать приложение» (рисунок 4.2):

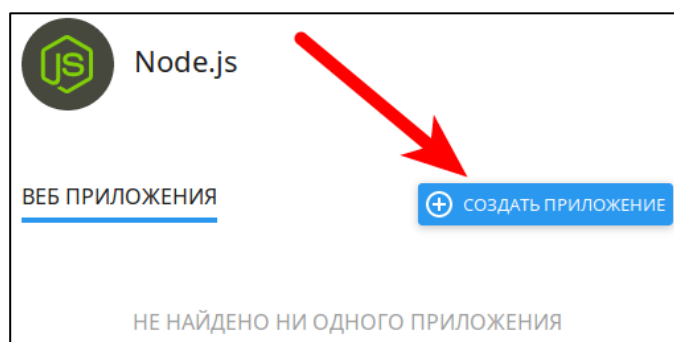


Рисунок 4.2 – Создание приложения

Необходимо выбрать требуемую версию Node.js, режим работы приложения (development или production), указать корневой каталог приложения (будет создан в корне вашего аккаунта), URL приложения (адрес, на котором будет запущено приложение), а также файл запуска приложения. После необходимо нажать на кнопку «Создать» (рисунок 4.3):

ВЕРБ ПРИЛОЖЕНИЯ **+** СОЗДАТЬ ПРИЛОЖЕНИЕ ОТМЕНИТЬ **СОЗДАТЬ**

Версия Node.js 12.9.0 ▾

Режим приложения Production ▾

Добавляет значение переменной NODE_ENV

Корневой каталог приложения test_app

Физический адрес приложения на сервере, соответствующий его URI

URL приложения yourdomain.by ▾

Ссылка (HTTP/HTTPS) на приложение

Файл запуска приложения index.js

Переменные окружения **+** ДОБАВИТЬ ПЕРЕМЕННУЮ

Рисунок 4.3 – Создание приложения

Если приложение использует базу данных MySQL, то необходимо создать её в cPanel в пункте «Мастер баз данных MySQL» раздела «Базы данных» (рисунок 4.4):

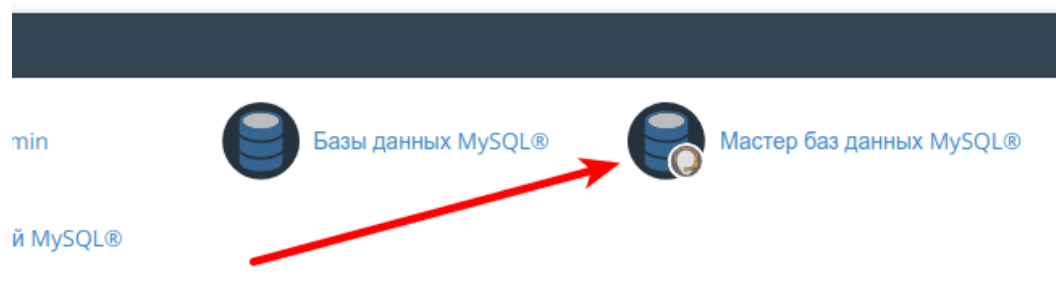


Рисунок 4.4 – Создание базы данных

Необходимо установить модули, требуемые для запуска приложения, указанные в файле `package.json` в директории с проектом (тут же при необходимости можно и отредактировать этот файл), нажатием на кнопку “Установить NPM пакеты” (рисунок 4.5):

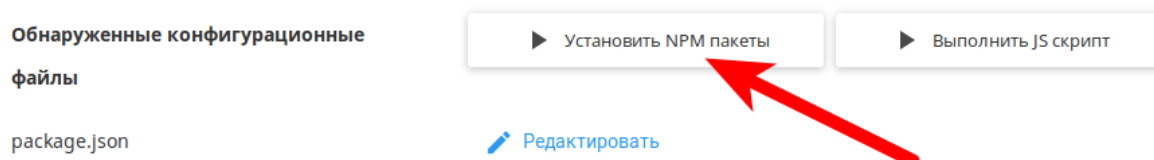


Рисунок 4.5 – Установка модулей

После установки всех необходимых модулей приложение будет доступно по адресу, заданному при создании.

4.2 Руководство пользователя

При переходе по адресу приложения, пользователь попадает на стартовое окно приложения. Стартовое окно изображено на рисунке 4.6.

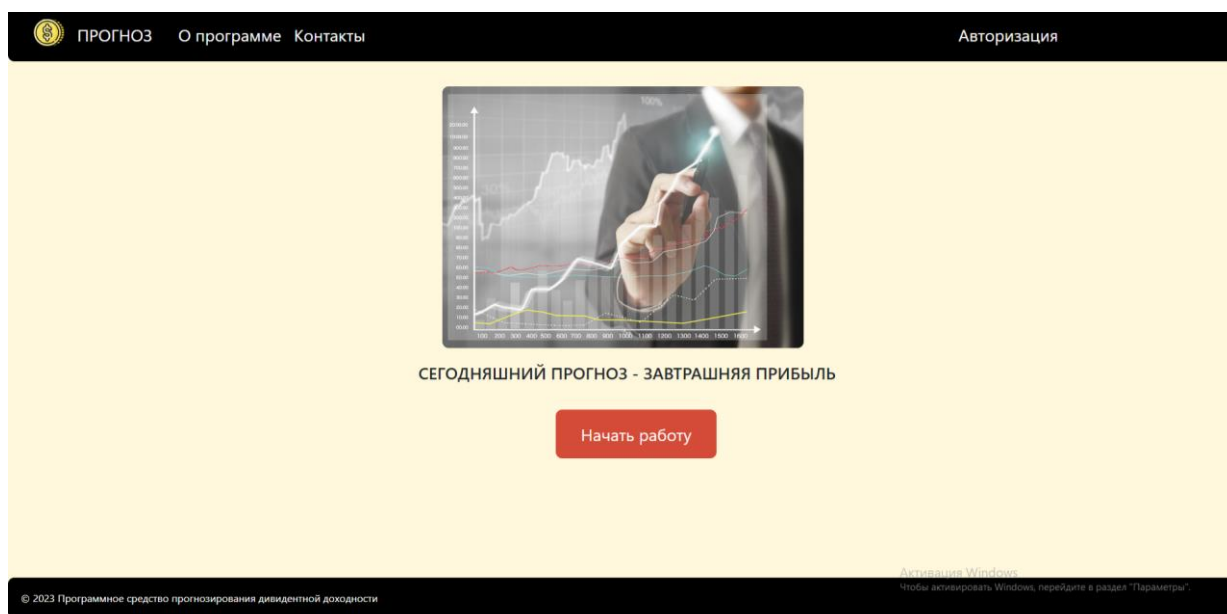


Рисунок 4.6 – Стартовое окно программного средства

При нажатии на ссылку «О программе» пользователь перейдет на вкладку, в которой изложены цели и задачи программного средства (рисунок 4.7):

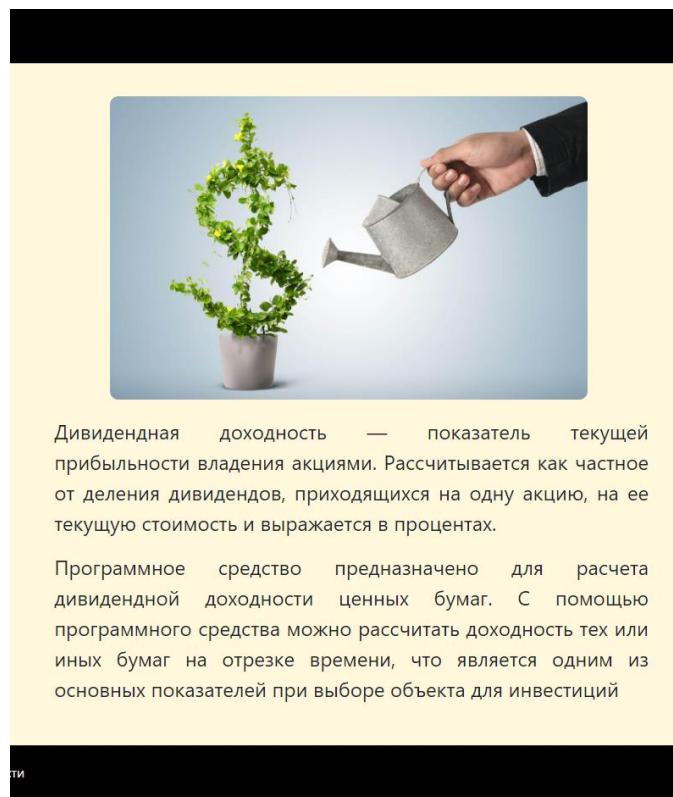


Рисунок 4.7 – Вкладка «О программе»

При переходе на вкладку «Контакты» пользователю будет доступна контактная информация (рисунок 4.8):

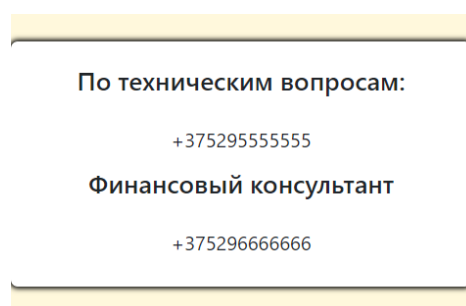


Рисунок 4.8 – Вкладка «Контакты»

Для авторизации в программном средстве, необходимо нажать кнопку «Авторизация», после чего заполнить предложенную форму (рисунок 4.9):

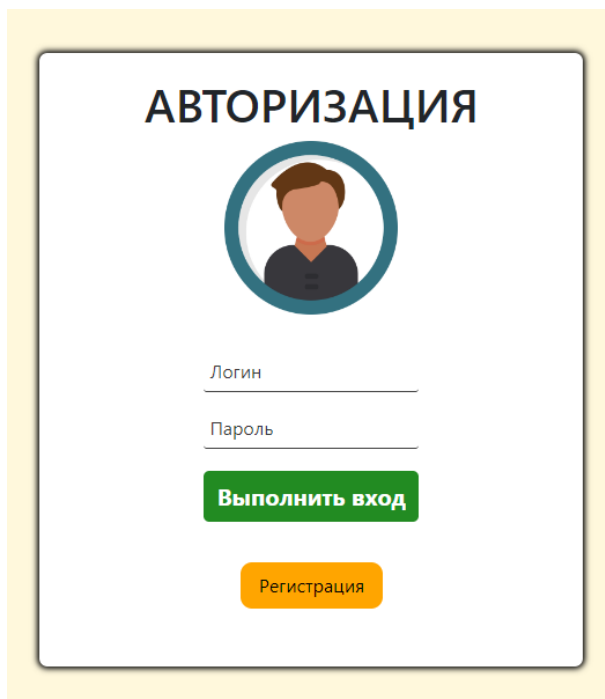


Рисунок 4.9 – Вкладка «Авторизация»

После прохождения авторизации пользователю доступно основное окно программы, которое представлено на рисунке 4.10:

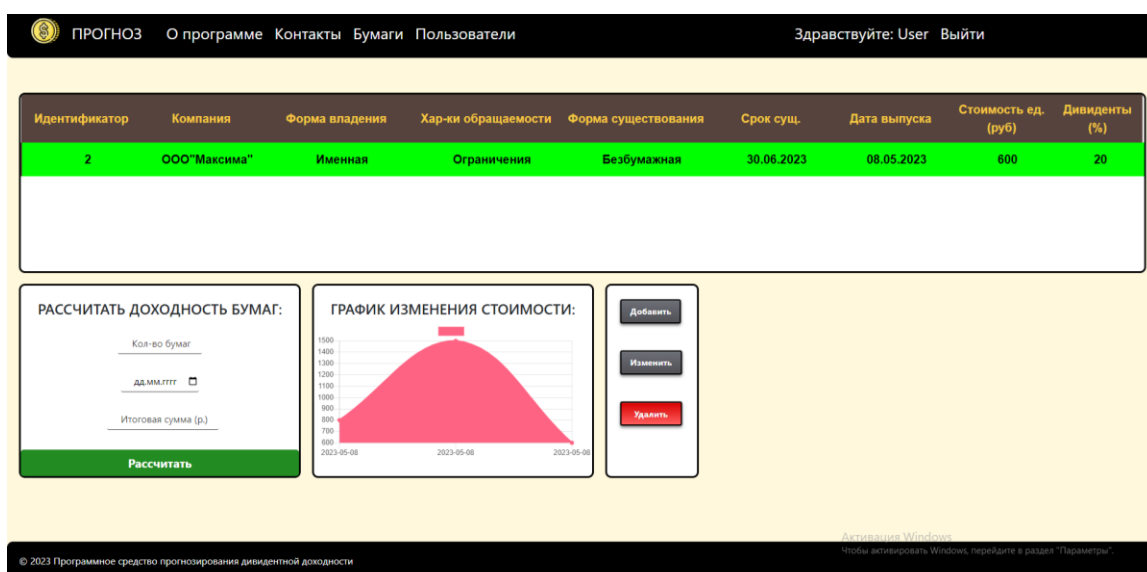
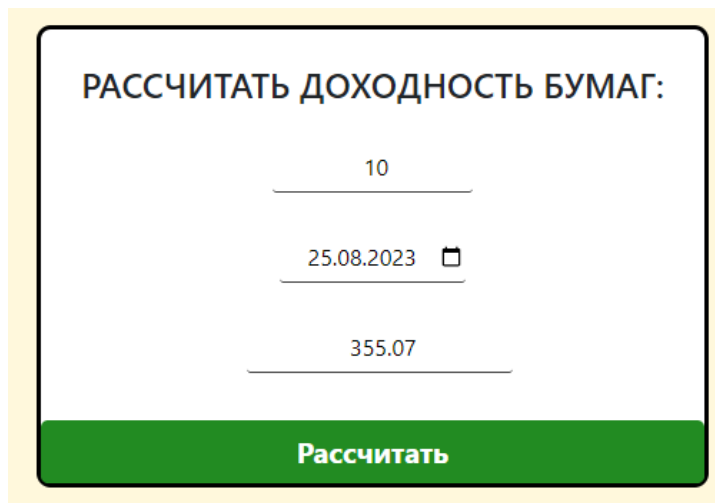


Рисунок 4.10 – Вкладка «Бумаги»

Вкладка «Бумаги» предоставляет пользователю возможность рассчитывать потенциальную прибыль при приобретении тех или иных активов (рисунок 4.11):



РАССЧИТАТЬ ДОХОДНОСТЬ БУМАГ:

10

25.08.2023

355.07

Рассчитать

Рисунок 4.11 – Поле для расчета доходности

Также основная рабочая вкладка позволяет просматривать график изменения стоимости тех или иных активов с отметкой точек изменения стоимости (рисунок 4.12):

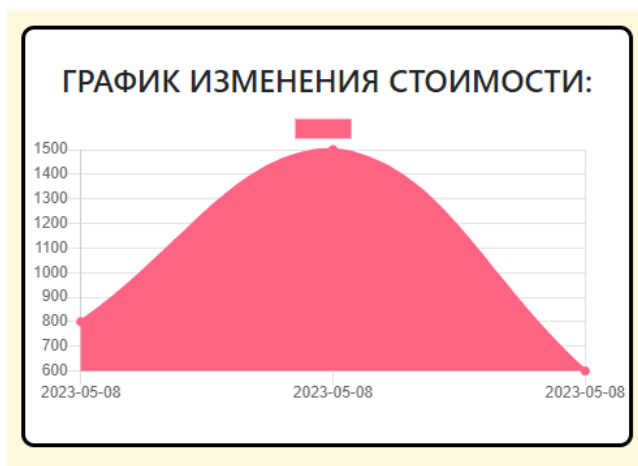


Рисунок 4.12 – График изменения стоимости

Для работы с пользователями необходимо перейти на вкладку «Пользователи» (рисунок 4.13):

Идентификатор	Имя	Роль	Изменение данных	Удаление
2	User4	Пользователь	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>
3	User	Администратор	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>
4	User2	Представитель	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>
5	User3	Пользователь	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>

Рисунок 4.13 – Вкладка «Пользователи»

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта решены следующие задачи.

Проведен анализ существующих программных средств для прогнозирования дивидендной доходности, показавший, что ни одно из них не удовлетворяет всем предъявляемым требованиям. На основании анализа выполнена постановка задачи на разработку программного средства для прогнозирования доходности ценных бумаг.

На основании проектной документации разработаны и программно реализованы алгоритмы функционирования программного средства. Программное средство реализовано на основе двухуровневой архитектуры: клиент и веб-сервер. Реализация выполнена с использованием технологии Node.js на языке программирования JavaScript. Результаты тестирования показали, что разработанное программное средство удовлетворяет функциональным требованиям и функции выполняются корректно.

Изложенное позволяет сделать вывод о том, что цель, поставленная в курсовой работе, достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Сиерра, К. Изучаем JavaScript / К. Сиерра, Б. Бейтс. – М.: Эксмо, 2020. – 720 с.
- [2] Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL / Л. Веллинг, Л. Томсон. – М.: Вильямс, 2018. – 848 с.
- [3] Bootstrap [Электронный ресурс]. – Режим доступа: <https://getbootstrap.com/docs/3.3/>. – Дата доступа: 31.03.2023.
- [4] CSS [Электронный ресурс]. – Режим доступа: <https://css.com>. – Дата доступа: 31.03.2023.
- [5] Новиков, Б. А. Настройка приложений баз данных / Б. А. Новиков, Г.Р. Домбровская. – СПб.: БХВ-Петербург, 2016. – 240 с.
- [6] Кузнецов, С. Д. Базы данных / С. Д. Кузнецов. – М.: Academia, 2016. – 496 с.
- [7] Илюшечкин, В. М. Основы использования и проектирования баз данных / В. М. Илюшечкин. – М.: Юрайт, 2016. – 214 с.
- [8] Тампре, Л. Введение в тестирование программного обеспечения / Л. Тампре. – Москва: Вильямс, 2020. – 368 с.
- [9] Майерс, Г. Искусство тестирования программ / Г. Майерс, Т. Баджетт, К. Сандлер. – Москва : Вильямс, 2019. – 272 с.

ПРИЛОЖЕНИЕ А

(обязательное)

Отчет о проверках на заимствование



Отчет о проверке на заимствования №1



Автор: [REDACTED] ID: 8968981

Проверяющий:

Отчет предоставлен сервисом «Антиплагиат» - <http://users.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 9
Начало загрузки: 09.05.2023 04:33:37
Длительность загрузки: 00:00:02
Имя исходного файла: ПЗ.pdf
Название документа: ПЗ
Размер текста: 65 кБ
Символов в тексте: 66846
Слов в тексте: 7756
Число предложений: 685

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Начало проверки: 09.05.2023 01:33:39
Длительность проверки: 00:00:04
Комментарии: не указано
Модуль поиска: Интернет Free



СОВПАДЕНИЯ

11,34%

САМОЦИТИРОВАНИЯ

0%

ЦИТИРОВАНИЯ

0%

ОРИГИНАЛЬНОСТЬ

88,66%

Совпадения - фрагменты проверяемого текста, полностью или частично сходные с найденными источниками, за исключением фрагментов, которые система отнесла к цитированию или самоцитированию. Показатель «Совпадения» - это доля фрагментов проверяемого текста, отнесенных к совпадениям, в общем объеме текста.

Самоцитирование - фрагменты проверяемого текста, совпадающие или почти совпадающие с фрагментом текста источника, автором или соавтором которого является автор проверяемого документа. Показатель «Самоцитирование» - это доля фрагментов текста, отнесенных к самоцитированию, в общем объеме текста.

Цитирования - фрагменты проверяемого текста, которые не являются авторскими, но которые система отнесла к корректно оформленным. К цитированиям относятся также шаблонные фразы: библиография; фрагменты текста, найденные модулем поиска «СПС Гарант: нормативно-правовая документация». Показатель «Цитирования» - это доля фрагментов проверяемого текста, отнесенных к цитированию, в общем объеме текста.

Текстовое пересечение - фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.

Источник - документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.

Оригинальный текст - фрагменты проверяемого текста, не обнаруженные ни в одном источнике и не отмеченные ни одним из модулей поиска. Показатель «Оригинальность» - это доля фрагментов проверяемого текста, отнесенных к оригинальному тексту, в общем объеме текста.

«Совпадения», «Цитирования», «Самоцитирование», «Оригинальность» являются отдельными показателями, отображаются в процентах и в сумме дают 100%, что соответствует полному тексту проверяемого документа.

Обращаем Ваше внимание, что система находит текстовые совпадения проверяемого документа с проиндексированными в системе источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности совпадений или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

№	Доля в тексте	Источник	Актуален на	Модуль поиска
[01]	5,07%	Пользовательский интерфейс http://studopedia.net	14 Ноя 2015	Интернет Free
[02]	3,67%	Базовые методы программной защиты http://lektii.org	24 Окт 2017	Интернет Free
[03]	3,67%	Базовые методы программной защиты — Информатика, информационные технологии http://csaa.ru	23 Июл 2019	Интернет Free

Активация
Чтобы активир
Еще источников: 5
Еще совпадений: 3,33%

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг кода алгоритмов

app.js

```
// Imports
const express = require('express')
const bodyParser = require('body-parser')
const expressLayouts = require('express-ejs-layouts')
const url = require('url')

const app = express()
const port = 5000

const func = require("./data/config.js");
const check = require("./data/checks.js");

// Static Files
app.use(express.static('public'))
// Example for other folders - not required
// app.use('/css', express.static(__dirname + 'public/css'))

// Set Templating Engine
app.use(expressLayouts)
app.set('layout', './layouts/full-width')
app.set('view engine', 'ejs')
var curUserName = "";
var curUserRole = "";

// GET
app.get('', (req, res) => {
    res.render('index', { title: 'Home Page', userName:curUserName,
    userRole:curUserRole})
})
```

```

app.get('/modifyUser', (req, res) => {
    var idUser = req.query.id;
    var loginUser = req.query.login;
    var messageUser = req.query.message;
    console.info(req.query.login);
    res.render('modifyUser', { title: 'Modify User', id:idUser,
login:loginUser,      message:messageUser,      userName:curUserName,
userRole:curUserRole})
})

```

```

app.get('/modifyPapersForm', (req, res) => {
    var date1 = req.query.date_in;
    date1 = date1.split(".");
    var date1M = date1.reverse().join("-");

    var date2 = req.query.date_out;
    date2 = date2.split(".");
    var date2M = date2.reverse().join("-");

    var id_ = req.query.id;
    var company_ = req.query.company;
    var property_form_ = req.query.property_form;
    var char_revers_ = req.query.char_revers;
    var existanse_form_ = req.query.existanse_form;
    var date_in_ = date1M;
    var date_out_ = date2M;
    var sum_ = req.query.sum;
    var procent_ = req.query.procent;
    res.render('ModifyPaper', { title: 'Modify Paper', idd:id_,
company:company_,      property_form:property_form_,
char_revers:char_revers_, existanse_form:existanse_form_,
    date_in:date_in_,      date_out:date_out_,      sum:sum_,
procent:procent_,userName:curUserName, userRole:curUserRole})
})

```

```

app.get('/_avto', (req, res) => {
    var name = req.query.userName;
    var role = req.query.userRole;
    res.render('index', { title: 'Home Page', userName:name,
userRole:role })
})

```

```

app.get('/paper', (req, res) => {
    func.getPapers("", function(result){
        papersBuf = result;
        if(papersBuf.length)
        {
            func.getModifyLogs("", function(result){
                divBuf = result;
                res.render('paper', { title: 'Papers Page', papers:
papersBuf,
                divBuff:divBuf,
                userName:curUserName,
userRole:curUserRole}))
            });
        }
        else
        {
            res.render('paper', { title: 'Papers Page', papers:
papersBuf, userName:curUserName, userRole:curUserRole})
        }
        //rest of your code goes in here
    });
})

```

```

app.get('/users', (req, res) => {
    func.getUsers("", function(result){
        usersBuf = result;
        if(usersBuf.length)
        {
            func.getRoles("", function(result){

```

```

        roleslist = result;
        if(roles.length)
        {
            console.info(roleslist.indexOf(0));
            res.render('users', { title: 'Home Page', users:
usersBuf, roles: roleslist, userName:curUserName,
userRole:curUserRole})
        }
        else
        {
            res.render('message', { title:'Message',data: "Таблица
ролей пуста"});
        }
    });
}
else
{
    res.render('message', { title:'Message',data: "Таблица
пользователей пуста"});
}
//rest of your code goes in here
});
})

app.get('/exit', (req, res) => {
    app.set('layout', './layouts/full-width');
    curUserName = "";
    curUserRole = 0;
    res.render('index', { title: 'Стартовая страница',
userName:curUserName, userRole:curUserRole})
})

app.get('/contacts', (req, res) => {
    console.info(curUserRole);

```



```

        res.render('contacts', { title: 'Контакты', userName:curUserName,
userRole:curUserRole})
    })

app.get('/about', (req, res) => {
    res.render('about', { title: 'О программе', userName:curUserName,
userRole:curUserRole})
})

app.get('/avt', (req, res) => {
    res.render('avt', { title: 'Avtorize',error:''})
})

app.get('/avt_error', (req, res) => {
    var errorMes = req.query.message;
    res.render('avt', { title: 'Avtorize',error:errorMes})
})

app.get('/reg', (req, res) => {
    res.render('reg', { title: 'Register'})
})

app.get('/message_reg', (req, res) => {
    var mes = req.query.message;
    res.render('message', { title:'Message',data: mes})
})

app.get('/add_paper', (req, res) => {
    var mes = req.query.message;
    res.render('AddPaper', { title:'Add Paper', userName:curUserName,
userRole:curUserRole})
})

//POST
const urlencodedParser = bodyParser.urlencoded({extended: false,});

```

```

app.post('/avt_input', urlencodedParser, function (request,response) {
    if (!request.body) return response.sendStatus(400);

    func.checkUser(request.body.username,request.body.password,
function(result){
    usersBuf = result;
    if(usersBuf.length)
    {
        curUserName = request.body.username;
        console.info(usersBuf[0].RoleId);

        curUserRole = usersBuf[0].RoleId;
        app.set('layout', './layouts/full-avto');
        response.redirect(url.format({
            pathname:"/_avto",
            query:{
                "userName":request.body.username,
                "userRole":usersBuf.RoleId
            }}}));
    }
    else
    {
        response.redirect(url.format({
            pathname:"/avt_error",
            query:{
                "message":"Неправильный логин или пароль"
            }}}));
    }
    //rest of your code goes in here
});
});

app.post('/reg_input', urlencodedParser, function (request,response) {

    if (!request.body) return response.sendStatus(400);

```

```

        if(check.checkPassword(request.body.password,request.body.passwo
rd_))
        {
            func.addUser(request.body.username,request.body.password,
function(result){
                res = result;
                response.redirect(url.format({
                    pathname:"/message_reg",
                    query:{
                        "message":"Регистрация пройдена успешно.
Осуществите вход под созданной учетной записью."
                    }}}));
            });
        }
        else
        {
            response.redirect(url.format({
                pathname:"/message_reg",
                query:{
                    "message":"Введенные пароли не совпадают между
собой."
                }}}));
        }

    });

app.post('/del_user', urlencodedParser, function (request,response) {

    if (!request.body) return response.sendStatus(400);
    console.info(request.body);
    func.deleteUser(request.body.id, function(result){
        res = result;
        if(result)
        {
            response.redirect("/users");
        }
    });
}

```

```

    }
    else
    {
        response.redirect(url.format({
            pathname: "/message_reg",
            query: {
                "message": "Ошибка в ходе удаления пользователя."
            })
        }));
    }
});

app.post('/modifyUserPost',      urlencodedParser,      function
(request,response) {
    if (!request.body) return response.sendStatus(400);
    response.redirect(url.format({
        pathname: "/modifyUser",
        query: {
            "id": request.body.idUser,
            "login": request.body.username,
            "message": ""
        })
    }));
});

app.post('/modifyUserPost2',      urlencodedParser,      function
(request,response) {
    if (!request.body) return response.sendStatus(400);
    console.info(request.body);
    func.updateUser(request.body.idUser,request.body.username,request.bod
y.role, function(result){
        res = result;
        if(result)
        {
            response.redirect(url.format({
                pathname: "/modifyUser",

```



```

        {
            response.redirect(url.format({
                pathname: "/message_reg",
                query: {
                    "message": "Ошибка в ходе сброса пароля."
                })
            }));
        }
    });
});

app.post('/addPaper', urlencodedParser, function (request, response) {
    if (!request.body) return response.sendStatus(400);
    response.redirect(url.format({
        pathname: "/add_Paper",
        query: {

        }
    }));
});

app.post('/addPaperDB', urlencodedParser, function (request, response)
{

    if (!request.body) return response.sendStatus(400);
    func.addPaper(request.body.company, request.body.property_form,
request.body.char_revers, request.body.existance_form,
request.body.date_in, request.body.date_out, request.body.sum,
request.body.procent, function(result){
        res = result;
        if(res)
        {
            response.redirect("/paper");

        }
        else
        {

```

```

        response.redirect(url.format({
            pathname: "/message_reg",
            query: {
                "message": "Ошибка добавления ценных бумаг."
            })
        }));
    }
});

});

app.post('/modifyPapersPost',      urlencodedParser,      function
(request,response) {
    if (!request.body) return response.sendStatus(400);
    console.info(request.body.sum);
    response.redirect(url.format({

        pathname: "/modifyPapersForm",
        query: {
            "id": request.body.id,
            "company": request.body.company,
            "property_form": request.body.property_form,
            "char_revers": request.body.char_revers,
            "existanse_form": request.body.existanse_form,
            "date_in": request.body.date_in,
            "id": request.body.id,
            "date_out": request.body.date_out,
            "sum": request.body.sum[0],
            "procent": request.body.procent
        })
    }));
});

app.post('/modifyPaperDB',      urlencodedParser,      function
(request,response) {

```

```

        if (!request.body) return response.sendStatus(400);
        func.addModifyLog(request.body.id,request.body.sum,
function(result){
    res = result;
    });

    func.modifyPaper(request.body.id,request.body.company,
request.body.property_form,
    request.body.char_revers,          request.body.existance_form,
request.body.date_in,    request.body.date_out,    request.body.sum,
request.body.procent, function(result){
    res = result;
    if(res)
    {
        response.redirect("/paper");

    }
    else
    {
        response.redirect(url.format({
            pathname: "/message_reg",
            query: {
                "message": "Ошибка изменения данных ценных бумаг."
            })
        }));
    }
    });

});

app.post('/deletePaper', urlencodedParser, function (request,response)
{

    if (!request.body) return response.sendStatus(400);
    console.info(request.body.id);
    func.deletePaper(request.body.id, function(result){

```



```

    res = result;
    if(result)
    {
        response.redirect("/paper");
    }
    else
    {
        response.redirect(url.format({
            pathname: "/message_reg",
            query: {
                "message": "Ошибка в ходе удаления ценных бумаг."
            })
        }));
    }
});
});

```

```

// Listen on Port 5000
app.listen(port, () => console.info(`App listening on port ${port}`))

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

```
CREATE TABLE `logs` (  
  `ID` int NOT NULL,  
  `UserId` int DEFAULT NULL,  
  `Message` varchar(45) DEFAULT NULL,  
  `Date` datetime DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
CREATE TABLE `modifylog` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `DATE` datetime DEFAULT NULL,  
  `Sum` int DEFAULT NULL,  
  `Paper_Id` int DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;  
  
CREATE TABLE `papers` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `Company` varchar(45) DEFAULT NULL,  
  `Property_form` varchar(45) DEFAULT NULL,  
  `Char_Revers` varchar(45) DEFAULT NULL,  
  `Existanse_Form` varchar(45) DEFAULT NULL,  
  `Date_In` datetime DEFAULT NULL,  
  `Date_Out` datetime DEFAULT NULL,  
  `Sum` varchar(45) DEFAULT NULL,  
  `Procent` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `roles` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `Name` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

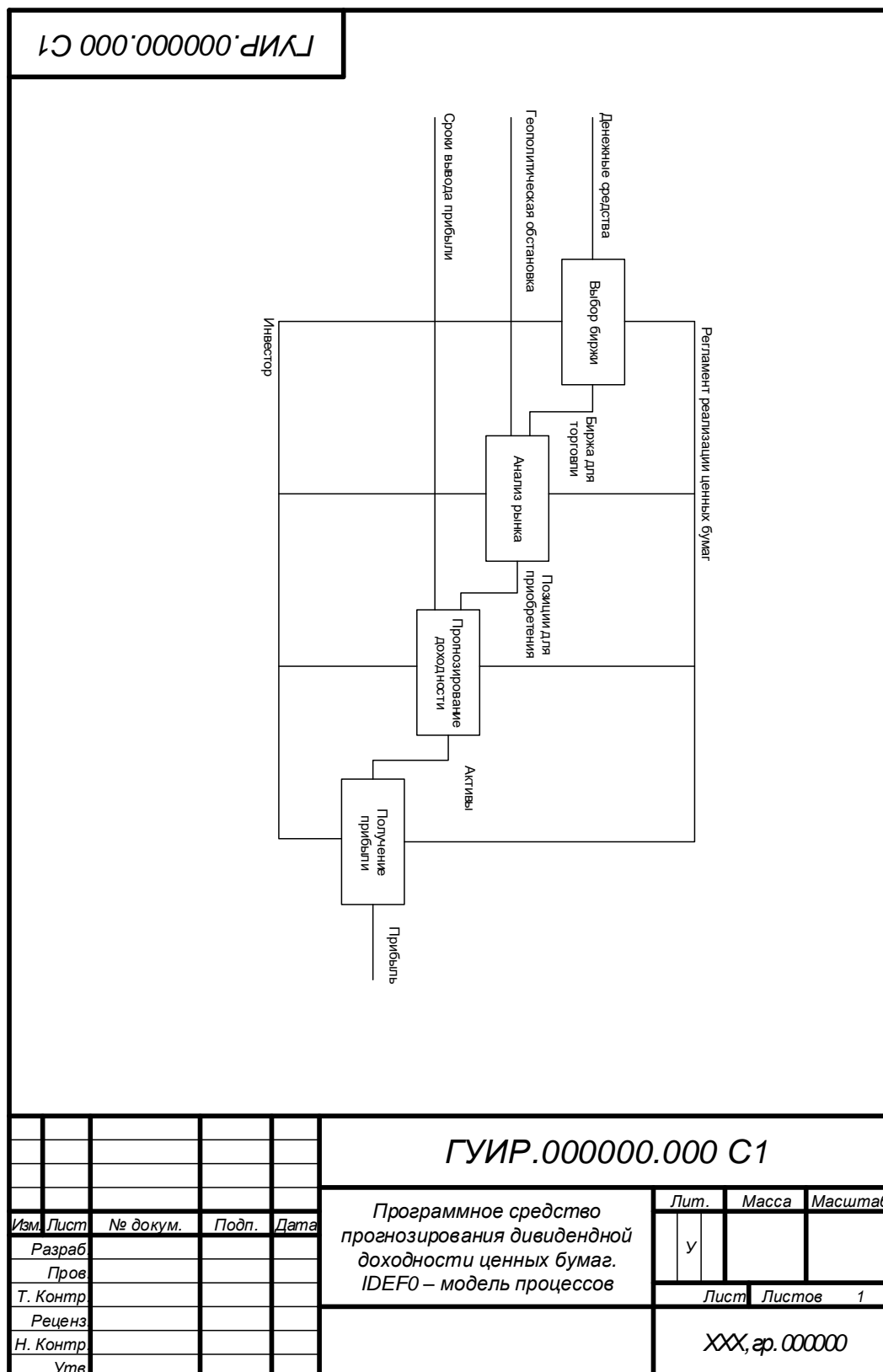
```
CREATE TABLE `users` (  
  `ID` int NOT NULL AUTO_INCREMENT,  
  `Login` varchar(45) DEFAULT NULL,  
  `Password` varchar(100) DEFAULT NULL,  
  `RoleId` int DEFAULT NULL,  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

(обязательное)

[illegible]

ПРИЛОЖЕНИЕ Д (обязательное)

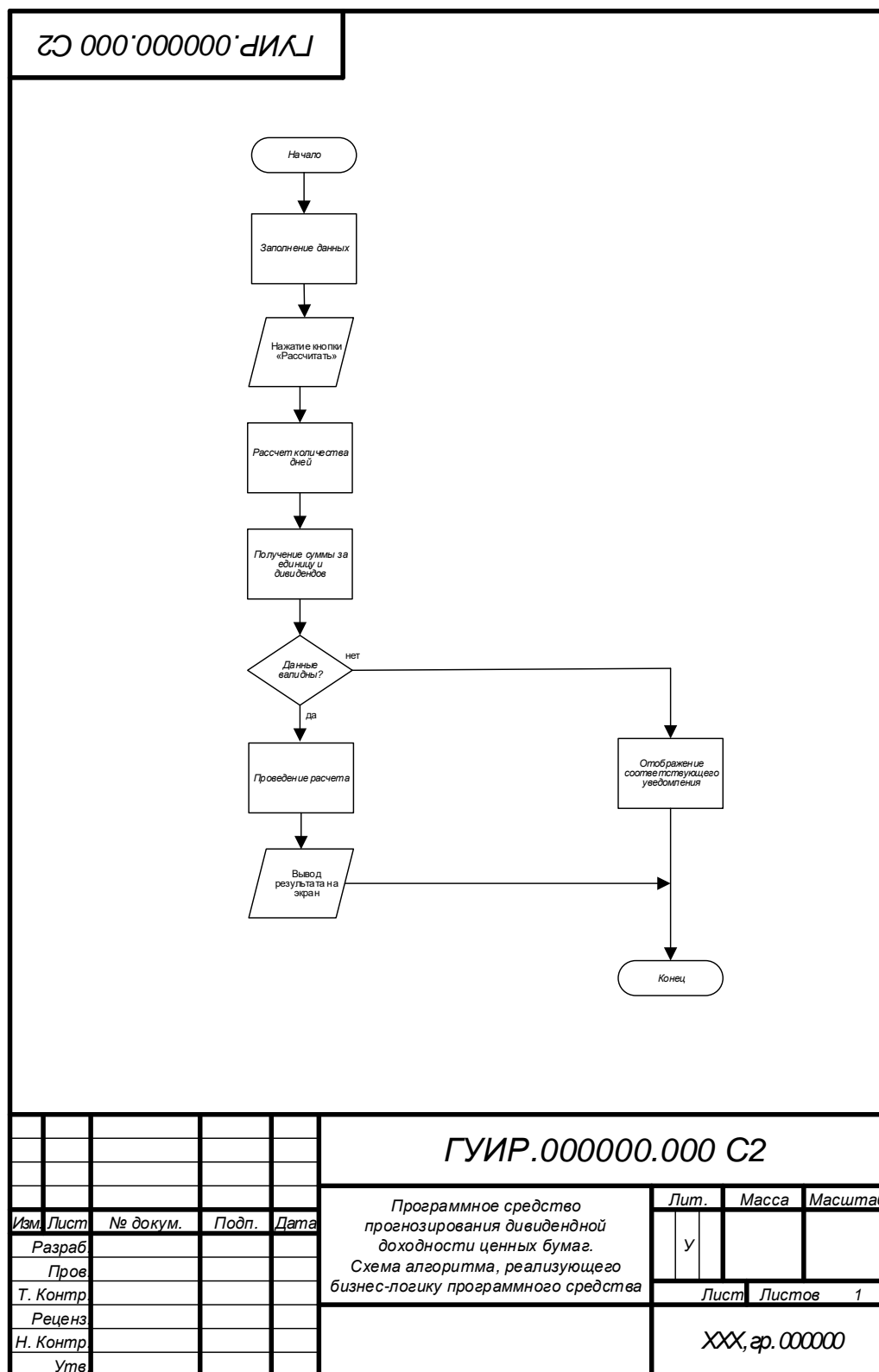
IDEF0-модель



ПРИЛОЖЕНИЕ Е

(обязательное)

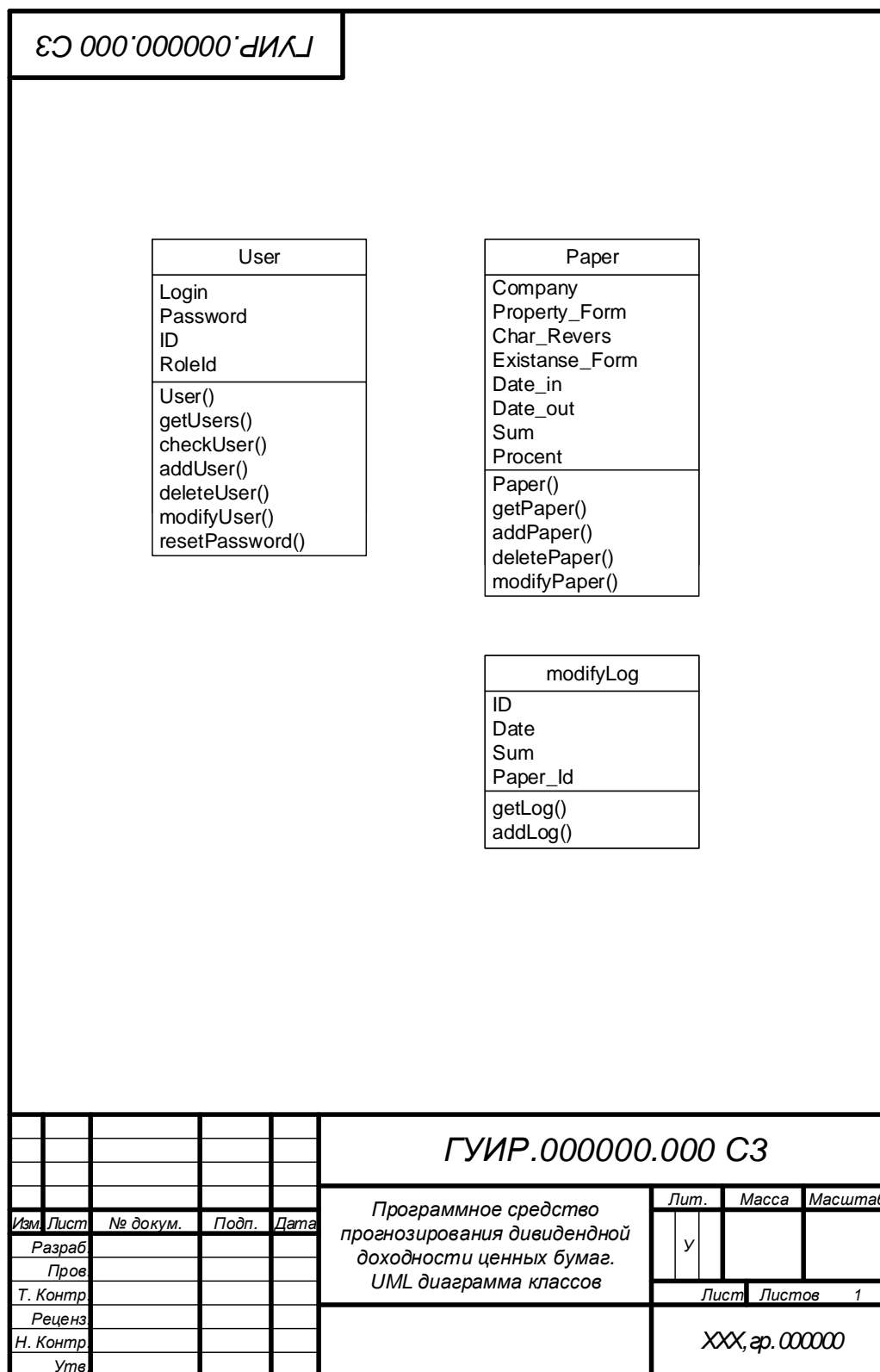
Схема алгоритма, реализующего бизнес-логику программного средства



ПРИЛОЖЕНИЕ Ж

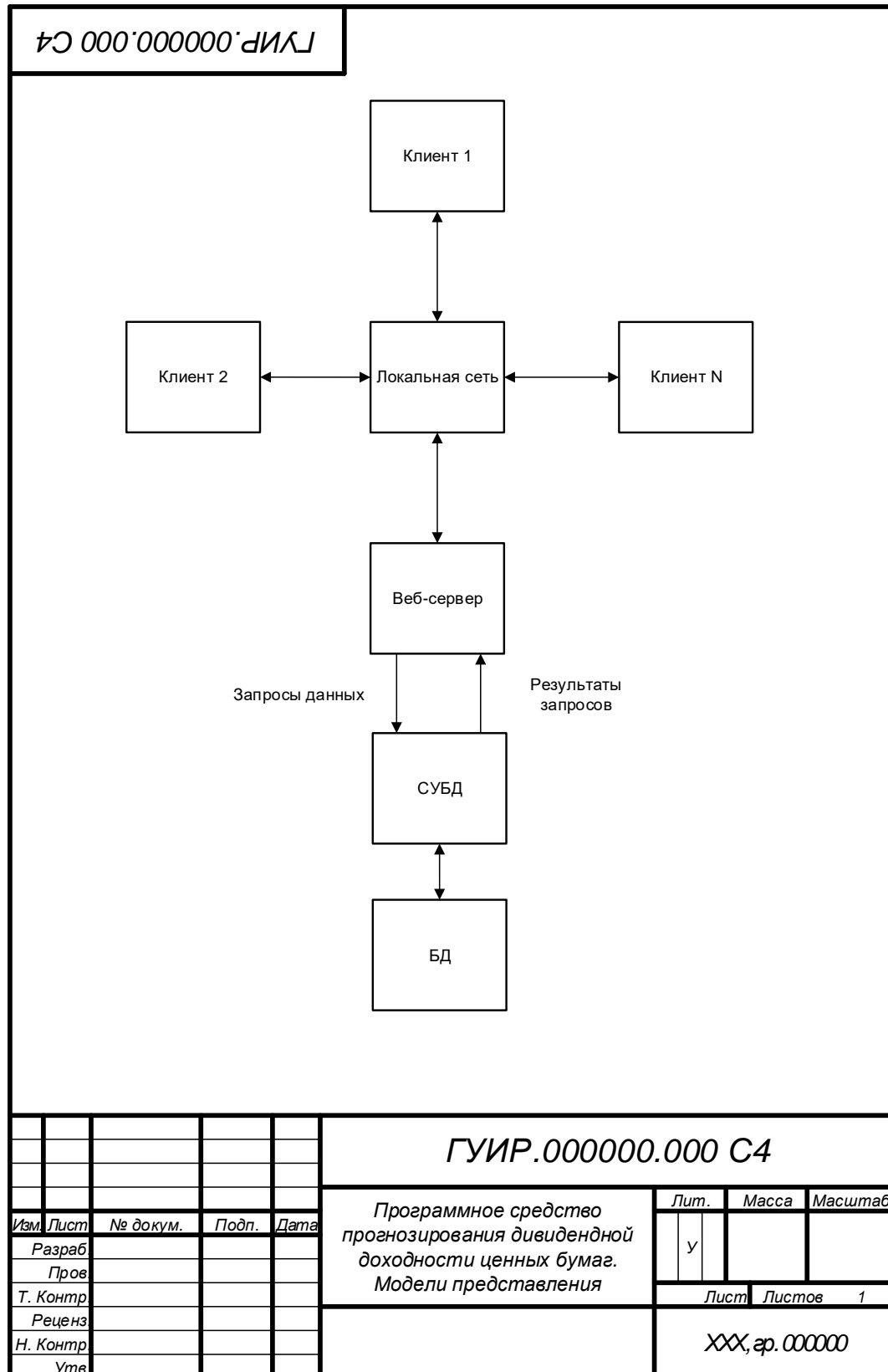
(обязательное)

UML диаграмма классов



ПРИЛОЖЕНИЕ 3 (обязательное)

Модель представления программного средства



Скриншоты рабочих столов

