Частное учреждение образования

«Колледж бизнеса и права»

Программное средство для автоматизации учёта работы сотрудника инвестиционной компании

Пояснительная записка

к курсовому проекту по дисциплине

«Конструирование программ и языки программирования»

КП Т.193021.401

Руководитель проекта ( Д. О. Елинская )

Обучающийся ( Р. М. Найдун )

2024

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.193021.401

Разраб.

Найдун Р.М

Провер.

Елинская Д.В

Реценз.

Н. Контр.

Утверд.

Программное средство для Автоматицазии учёта работы сотрудника инвестиционной компании

Лит.

Листов

КБП

49

у

**Содержание**

[Введение 4](#_Toc169507100)

[1 Описание задачи 5](#_Toc169507101)

[1.1 Анализ предметной области 5](#_Toc169507102)

[1.2 Постановка задачи 6](#_Toc169507103)

[2 Проектирование системы 7](#_Toc169507104)

[2.1 Требования к приложению 7](#_Toc169507105)

[2.2 Проектирование модели 8](#_Toc169507106)

[2.3 Концептуальный прототип 9](#_Toc169507107)

[3 Описание реализации программного средства 11](#_Toc169507108)

[3.1 Инструменты разработки и применяемые технологии 11](#_Toc169507109)

[3.2 Порядок авторизации пользователей 11](#_Toc169507110)

[3.3 Организация данных 12](#_Toc169507111)

[3.4 Функции: логическая и физическая организация 13](#_Toc169507112)

[3.5 Входные и выходные данные 16](#_Toc169507113)

[3.6 Функциональное тестирование 16](#_Toc169507114)

[3.7 Описание справочной системы 17](#_Toc169507115)

[4 Применение 18](#_Toc169507116)

[4.1 Назначение программного средства 18](#_Toc169507117)

[4.2 Условия применения 18](#_Toc169507118)

[Заключение 20](#_Toc169507119)

[Список информационных источников 21](#_Toc169507120)

[Приложение А (обязательное) 22](#_Toc169507121)

[Приложение Б (справочное) Формы входных и выходных документов 42](#_Toc169507122)

[Приложение В (обязательное) Результаты работы программы 43](#_Toc169507123)

# Введение

Современные инвестиционные компании сталкиваются с многочисленными вызовами, связанными с эффективным управлением и обработкой больших объемов данных. Эти данные включают информацию о сотрудниках, клиентах, инвестиционных портфелях и ценных бумагах. Проблемы, возникающие в процессе управления, могут существенно влиять на эффективность компании и её конкурентоспособность на рынке.

Описание задачи подробно описывает потребности пользователей и определяет требования к программному средству. В системе реализована роль «Клиент». Для хранения информации о продуктах и запасах используется база данных SQL. Клиент может составлять инвестиционные портфели, выбирать активы и автоматически рассчитывать стоимость портфеля. Система также предупреждает о необходимости пополнения баланса пользователя в случае нехватки.

В данной записке рассматривается выбор технологий для разработки ПС для управления программой для инвестирования «NorthVest», обосновывается целесообразность использования языка C# и базы данных MS SQL, учитывая требования к ПС, такие как быстрое выполнение операций, удобный интерфейс, надежность и безопасность данных, возможность создания отчетов и аналитических инструментов. Также приводятся примеры возможных решений с использованием данных технологий.

Введение описывает цели и задачи проекта, а также его актуальность. Описание задачи детализирует пользовательские потребности и определяет требования к программному обеспечению. Проектирование системы включает в себя создание архитектуры разрабатываемого ПО, а также разработку алгоритмов и методов для его планирования. Описание реализации программного обеспечения обосновывает выбор языка программирования и используемых инструментов, описывает ключевые модули и функции программы, а также процесс тестирования и отладки. Применение подробно рассматривает возможности и функциональность разработанного ПО, предоставляет примеры его использования и результаты работы. Заключение подводит итоги проекта, оценивает соответствие достигнутых результатов поставленным целям и задачам, выделяет преимущества разработанного ПО и обсуждает его перспективы. Список использованных источников включает литературу и другие ресурсы, использованные в процессе выполнения проекта.

Таким образом, данная записка представляет собой полный обзор проекта разработки программного средства для управления портфелями в программе «NorthVest», включая выбор технологий, проектирование системы, описание реализации, применение и заключение.

# Описание задачи

## Анализ предметной области

Темой курсового проекта является разработка «Программного средства для автоматизации учёта работы сотрудников инвестиционной компании «NorthVest».

Целью данного исследования является создание программного обеспечения, которое упростит и ускорит процесс управления данными о пользователях, инвестиционных портфелях и ценных бумагах в инвестиционной компании.

Основные бизнес-процессы, происходящие в компании, включают:

1. Управление пользователями. Пользователи могут регистрироваться и входить в систему с использованием логина и пароля. Программное средство позволяет создавать, изменять и удалять данные о пользователях.
2. Управление портфелями. Программное средство предоставляет возможность создания инвестиционного портфеля с указанием клиента, выбранных ценных бумаг и даты оформления.

Анализ данных. Программное средство собирает и анализирует данные о пользователях и инвестиционных портфелях, предоставляя возможность формирования статистических отчетов и графиков.

Алгоритмы получения промежуточных и итоговых показателей в программном средстве для управления инвестиционными портфелями включают:

1. Получение данных о портфелях. Программное средство получает информацию о портфелях (состав, баланс, изменения стоимости).
2. Визуализация данных. После анализа данных программное средство предоставляет пользователю возможность визуализировать эти показатели в виде графиков и отчетов для выявления трендов в изменении баланса и эффективности портфелей за выбранный промежуток времени.

Программное средство для управления инвестиционными портфелями обеспечивает автоматизацию процессов, связанных с обработкой информации о пользователях, портфелях и ценных бумагах, а также анализом данных. Это позволяет значительно ускорить и упростить работу персонала, снизить вероятность ошибок и повысить эффективность управления инвестициями.

Кроме того, компьютерная обработка информации позволяет получать более точные и полные данные о портфелях, а также выявлять тенденции и тренды в инвестициях. Это позволяет управляющим принимать более обоснованные решения, определять наиболее эффективные стратегии инвестирования и направления развития компании, а также улучшать качество обслуживания клиентов.

Таким образом, программное средство для автоматизации учёта работы сотрудников инвестиционной компании является необходимым инструментом для эффективного управления и повышения конкурентоспособности на рынке.

Потенциальными пользователями программного средства для управления инвестиционными портфелями могут быть владельцы инвестиционных компаний, финансовые аналитики и другие специалисты в сфере инвестиций. Также это программное средство может быть полезно менеджерам и администраторам инвестиционных компаний, занимающимся организацией работы персонала и управлением инвестициями. Кроме того, программное средство будет удобно и для клиентов, позволяя им получать подробную информацию о состоянии своих инвестиций и принимать более обоснованные решения.

## Постановка задачи

Для автоматизации программного средства для учёта работы сотрудников инвестиционной компании «NorthVest» необходимо определить следующие функции:

1. Управление пользователями. Система должна позволять пользователям регистрироваться, входить в систему с использованием логина и пароля, а также управлять своими данными. Администраторы должны иметь возможность изменять и удалять данные пользователей.
2. Управление портфелями. Система должна позволять создавать, редактировать и удалять инвестиционные портфели, указывая клиента, выбранные ценные бумаги и дату оформления. Программное средство должно автоматически обновлять информацию о балансе портфеля и его изменениях.
3. Анализ и отчётность. Система должна предоставлять отчеты о состоянии портфелей, изменениях баланса и эффективности инвестиций. Эти данные помогут аналитикам и управляющим принимать обоснованные решения для улучшения инвестиционных стратегий и оптимизации процессов.

Одним из аналогов программного средства для управления инвестиционными портфелями является платформa Interactive Brokers - https://www.interactivebrokers.com/. В данном программном обеспечении представлена возможность управления инвестициями, включая покупку и продажу ценных бумаг, анализ рыночных данных и отчеты по портфелям. Отличие в том, что Interactive Brokers предлагает интеграцию с различными торговыми платформами.

Еще один аналог - Charles Schwab - https://www.schwab.com/. В этом программном обеспечении также есть функции управления инвестициями, анализ рыночных данных и отчеты по портфелям. Отличие заключается в том, что Charles Schwab предоставляет специализированные услуги для различных типов инвесторов, включая индивидуальные и институциональные клиенты, а также предлагает обширную образовательную поддержку.

Разрабатываемое программное средство для управления инвестициями в компании «NorthVest» будет отличаться от этих платформ тем, что оно предназначено для внутреннего использования в компании, ориентировано на работу с данными о клиентах, портфелях и ценных бумагах.

# Проектирование системы

## Требования к приложению

В процессе работы программного средства для учёта работы сотрудников инвестиционной компании «NorthVest» возможны различные риски, которые могут привести к нарушению безопасности данных клиентов и сотрудников. Чтобы избежать этих рисков, необходимо использовать средства защиты.

Средства защиты для данного программного средства могут быть различными, их выбор зависит от конкретных условий работы. Однако можно выделить несколько основных средств защиты, которые должны быть использованы в любом случае:

1. **Антивирусное программное обеспечение**. Это критически важное средство защиты от вредоносных программ и вирусов. Для системы «NorthVest» необходимо установить надежное антивирусное ПО и регулярно обновлять его.
2. Пароли и аутентификация. Для обеспечения безопасности системы управления инвестициями необходимо использовать сложные пароли и методы аутентификации. Это позволит контролировать доступ к системе и защитить её от несанкционированного доступа.

При разработке данного программного средства необходимо учитывать ряд факторов, которые могут повлиять на его эффективность и удобство использования. Один из таких факторов – ограничения, которые накладываются на интерфейс для пользователей.

Первое ограничение, которое необходимо учесть при разработке системы управления инвестициями, – это доступность для всех категорий пользователей. Программное средство «NorthVest» будет использоваться людьми разного возраста и уровня образования, поэтому интерфейс должен быть интуитивно понятным и простым в использовании. Необходимо избегать сложных терминов и технических терминов, а также обеспечить возможность быстрого обучения новым пользователям.

Второе ограничение – это ограничение доступа к функциям системы. Необходимо создать интерфейс, который будет позволять пользователям выполнять только те функции, которые относятся к их ролям и правам доступа. Например, аналитики должны иметь доступ только к функциям, связанным с анализом данных, а администраторы – к функциям, связанным с управлением пользователями и портфелями.

Третье ограничение – это ограничение на количество информации, отображаемой на одном экране. Интерфейс должен быть удобным для чтения и использования, что означает, что он не должен быть перегружен информацией. Необходимо определить основные функции и информацию, которые будут отображаться на главном экране, а дополнительную информацию можно предоставлять по запросу пользователя.

Интерфейс программного средства для учёта работы сотрудников инвестиционной компании «NorthVest» должен быть интуитивно понятным и простым в использовании, так как он будет использоваться персоналом, который может иметь различный уровень компьютерной грамотности. Ниже описаны общие требования к интерфейсу программного средства:

1. Расположение элементов управления должно быть логичным и понятным.
2. Размер элементов управления должен быть достаточным для удобного использования. Кнопки и текстовые поля должны быть достаточно большими, чтобы их можно было легко прочитать и нажать.
3. **Форма элементов управления** должна быть понятной. Например, кнопка для подтверждения создания портфеля должна иметь соответствующий значок или текст, а кнопка для отмены действия должна быть обозначена символом крестика или текстом «Отмена».
4. Шрифт, используемый в интерфейсе, должен быть читаемым и понятным. Рекомендуется использовать шрифты с засечками, такие как Times New Roman или Arial, чтобы обеспечить лучшую читаемость.

Для программного средства «NorthVest» требуются определенные аппаратные и операционные ресурсы.

Операционная система: для работы ПС необходима операционная система Windows 7 или более новая версия.

Объем памяти устройства: рекомендуется иметь не менее 4 Гб оперативной памяти для более эффективной работы программы.

Дополнительные устройства: для управления инвестициями в компании «NorthVest» необходимы мышь, клавиатура и принтер. Мышь и клавиатура используются для ввода информации, а принтер – для печати отчетов.

## Проектирование модели

Диаграмма вариантов использования позволяет понять, как происходит управление инвестиционными портфелями в компании «NorthVest». Она помогает понять процесс работы программного средства и выявить возможные ошибки в его функционировании. Эта диаграмма представлена на графической части, на листе 1.

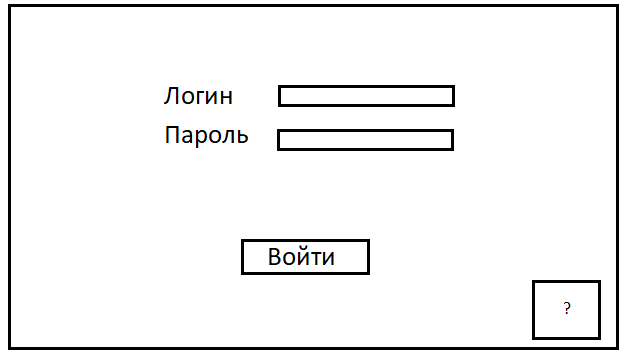
Диаграмма классов — это структурная диаграмма UML, показывающая общую структуру иерархии классов системы, а также их взаимосвязи, атрибуты, методы и интерфейсы. Эта диаграмма широко используется для документирования, визуализации и создания систем путем прямого и обратного проектирования. Диаграмма классов для программного средства «NorthVest» показана на графической части, на листе 2.

Диаграмма последовательности может быть использована для того, чтобы помочь сотрудникам понять процессы, происходящие в программе, и причины возникновения ошибок. Эта диаграмма показана на графической части, на листе 3.

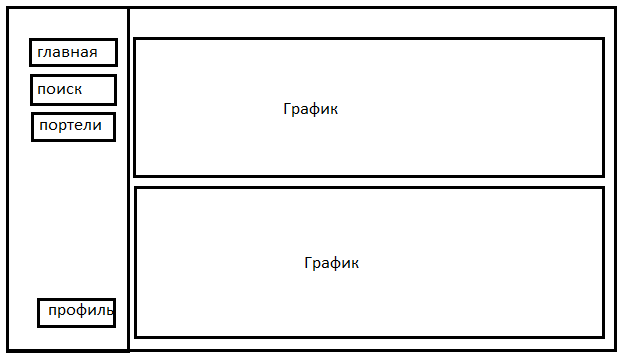
Диаграмма деятельности используется для анализа и оптимизации бизнес-процессов, планирования работы, принятия решений и управления проектами. Эта диаграмма показана на графической части, на листе 4.

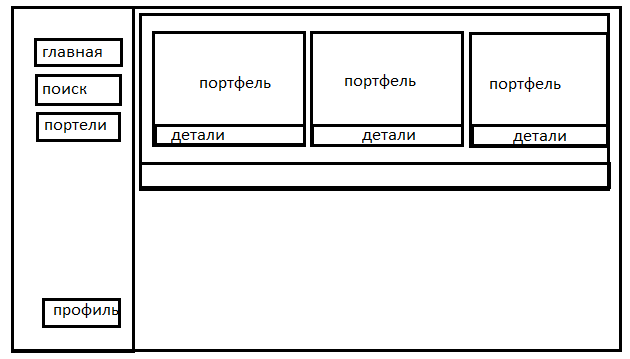
## Концептуальный прототип

При запуске программного средства «NorthVest» пользователю будет предложено ввести логин и пароль для подтверждения личности. Если введены неверные данные, переход к следующему окну не осуществится, и система запросит данные повторно. Если данные верны, пользователь переходит к окну профиля, где отображаются все функции, доступные данному пользователю. Также любой пользователь может ознакомиться с программой; при нажатии на кнопку «?» открывается справочник по программе. Макет формы «Авторизация» представлен на рисунке 1.

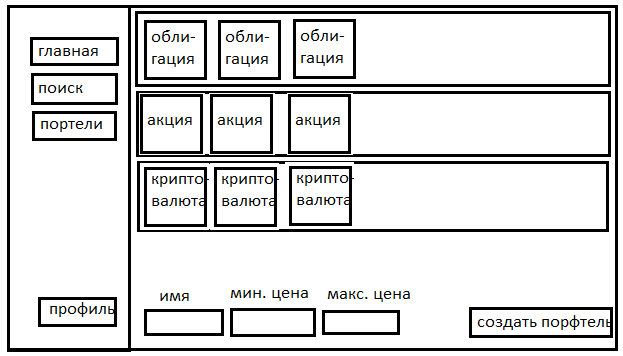


Главное окно ПС «NorthVest» должно быть удобным и интуитивно понятным. На экране должны быть размещены кнопки для доступа к профилю пользователя, портфелям, доступным ценным бумагам. Также должно быть поле с текущей информацией о пользователе и его портфелях, чтобы пользователь мог в любое время посмотреть состояние своих активов. Макет формы «Главная» представлен на рисунке 2. После выбора инвестиционного портфеля открывается окно с подробной информацией о выбранных ценных бумагах и их текущей стоимости. Здесь также показывается общая сумма портфеля. При необходимости пользователь может распечатать отчет о состоянии портфеля. Макет формы «Портфели» представлен на рисунке 3





При открытии окна «Поиск» происходит показ доступных активов. Возможно добавление активов в портфель. Макет формы «Поиск» представлен на рисунке 4.



# Описание реализации программного средства

## Инструменты разработки и применяемые технологии

C# отличается своей мощностью и гибкостью, что делает его идеальным выбором для создания широкого спектра приложений, от игр до комплексных систем управления и инструментов для анализа данных.

В контексте разработки программного средства (ПС) для автоматизации учёта работы сотрудников инвестиционной компании «NorthVest», C# может быть использован благодаря следующим преимуществам:

* Высокая скорость обработки: C# обеспечивает быстродействие, необходимое для эффективного управления и обработки данных пользователей и их инвестиционных портфелей;
* Интуитивно понятный пользовательский интерфейс: Использование графических библиотек C# позволяет создать привлекательный и понятный интерфейс для пользователей;
* Защита данных: Интеграция с системами баз данных, такими как MS SQL, гарантирует безопасное хранение и управление информацией о пользователях, портфелях и ценных бумагах;
* Аналитика и отчетность: C# в сочетании с базой данных позволяет разрабатывать сложные отчеты и аналитические инструменты для мониторинга и анализа финансовых данных.

Также, при разработке ПС для «NorthVest» следует учесть следующие аспекты:

* Оперативность: ПС должно обеспечивать мгновенную реакцию на действия пользователей, такие как просмотр и обновление данных;
* Дружелюбный интерфейс: Интерфейс должен быть интуитивно понятным и простым, чтобы новые пользователи могли легко освоить его без длительного обучения;
* Надежность: Система должна быть устойчива к сбоям и защищена от несанкционированного доступа;
* Гибкость отчетов: ПС должно предоставлять возможности для генерации персонализированных отчетов по запросу пользователя или управляющего.

Использование C# и MS SQL в разработке ПС для «NorthVest» обеспечит не только техническую эффективность, но и удобство в эксплуатации, что является ключевым фактором для успешной работы инвестиционной компании.

## Порядок авторизации пользователей

Клиент должен авторизоваться в системе с помощью своих учетных данных (логин и пароль), а если он ранее не использовал данное ПО, то ему следует зарегистрироваться с помощью ввода своих учетных данных (логин и пароль). Он может добавлять активы в портфель, продавать его, редактировать.

## Организация данных

Таблица «userD» хранит информацию о пользователях, структура приведена в таблице 3.1.

Таблица 3.1 – Структура таблица «userD»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| userId | Int | 4 | Id пользователя |
| login | nvarchar | 10 | Логин |
| pass | nvarchar | 10 | Пароль |
| fio | varchar | 50 | ФИО |
| userBal | decimal | 18 | Баланс |

Таблица «Assets» хранит информацию о активах, структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблица «Assets»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| AssetId | Int | 4 | Id актива |
| Name | nvarchar | 100 | Название актива |
| Type | nvarchar | 50 | Тип актива |
| Price | decimal | 18 | Цена актива |

Таблица «Portfolios» хранит информацию о портфелях, структура приведена в таблице 3.3.

Таблица 3.3 – Структура таблица «Portfolios»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| PortfolioId | Int | 4 | Id портфеля |
| UserId | Int | 4 | Id пользователя |
| PortfolioName | nvarchar | 100 | Название портфеля |
| CreatedDate | datetime | 8 | Дата создания портфеля |

Таблица «PortfolioDetails» хранит информацию о деталях портфеля, структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблица «PortfolioDetails»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| PortfolioDetailId | Int | 4 | Id деталей |

Продолжение таблицы 3.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| PortfolioId | Int | 4 | Id портфеля |
| AssetId | Int | 4 | Id актива |
| Quantity | Int | 4 | Количество |

Таблица «AssetPriceHistory» хранит информацию о истории цен активов, структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблица «AssetPriceHistory»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| AssetPriceHistoryId | Int | 4 | Id истории |
| AssetId | Int | 4 | Id актива |
| Price | decimal | 18 | Цена актива |
| RecordedDate | datetime | 8 | Дата изменения цены |

Таблица «PortfolioValueHistory» хранит информацию о истории цен портфелей, структура приведена в таблице 3.6.

Таблица 3.6 – Структура таблица «PortfolioValueHistory»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| PortfolioValueHistoryId | Int | 4 | Id истории |
| рId | Int | 4 | Id портфеля |
| Value | decimal | 18 | Цена портфеля |
| RecordedDate | datetime | 8 | Дата изменения цены |

## Функции: логическая и физическая организация

Основные функции программного средства представлены ниже.

«Добавление активов в портфель». Входные параметры: вид актива(String), наименование (String), количество (int), цена (double). Выходные параметры: Отсутствуют. Пример программного кода:

private void CreatePortfButton3\_Click(object sender, EventArgs e)

{

if (selectedAssets.Count == 0)

{

MessageBox.Show("Please select at least one asset for the portfolio.");

return;

}

string portfolioName = GetPortfolioName();

if (string.IsNullOrEmpty(portfolioName))

{

MessageBox.Show("Please provide a valid name for the portfolio.");

return;

}

decimal totalCost = 0;

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

using (SqlTransaction transaction = con.BeginTransaction())

{

try

{

foreach (var asset in selectedAssets)

{

SqlCommand priceCmd = new SqlCommand("SELECT Price FROM Assets WHERE AssetId = @AssetId", con, transaction);

priceCmd.Parameters.AddWithValue("@AssetId", asset.Key);

decimal price = (decimal)priceCmd.ExecuteScalar();

totalCost += price \* asset.Value;

}

decimal currentBalance = GetUserBalance();

if (currentBalance < totalCost)

{

MessageBox.Show("Insufficient balance to purchase selected assets.");

transaction.Rollback();

return;

}

SqlCommand cmd = new SqlCommand("INSERT INTO Portfolios (UserId, PortfolioName) OUTPUT INSERTED.PortfolioId VALUES (@UserId, @PortfolioName)", con, transaction);

cmd.Parameters.AddWithValue("@UserId", userId);

cmd.Parameters.AddWithValue("@PortfolioName", portfolioName);

int portfolioId = (int)cmd.ExecuteScalar();

foreach (var asset in selectedAssets)

{

SqlCommand insertCmd = new SqlCommand("INSERT INTO PortfolioDetails (PortfolioId, AssetId, Quantity) VALUES (@PortfolioId, @AssetId, @Quantity)", con, transaction);

insertCmd.Parameters.AddWithValue("@PortfolioId", portfolioId);

insertCmd.Parameters.AddWithValue("@AssetId", asset.Key);

insertCmd.Parameters.AddWithValue("@Quantity", asset.Value);

insertCmd.ExecuteNonQuery();

}

decimal newBalance = currentBalance - totalCost;

UpdateUserBalance(newBalance);

string insertTransactionSql = "INSERT INTO UserTransactions (UserId, TransactionType, Amount, TransactionDate) VALUES (@UserId, @TransactionType, @Amount, @TransactionDate)";

SqlCommand insertTransactionCmd = new SqlCommand(insertTransactionSql, con, transaction);

insertTransactionCmd.Parameters.AddWithValue("@UserId", userId);

insertTransactionCmd.Parameters.AddWithValue("@TransactionType", "Трата");

insertTransactionCmd.Parameters.AddWithValue("@Amount", totalCost);

insertTransactionCmd.Parameters.AddWithValue("@TransactionDate", DateTime.Now);

insertTransactionCmd.ExecuteNonQuery();

transaction.Commit();

MessageBox.Show("Portfolio created successfully.");

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show("Ошибка при создании портфеля: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

«Создание отчёта». Входные параметры: количество дней (int), цена (double), дата изменения цены (DateTime). Выходные параметры: отчёт в формате .xlsx . Пример программного кода:

public partial class PortfolioDetailsForm : Form

{

private int portfolioId;

private int userId;

public PortfolioDetailsForm(int portfolioId, int userId)

{

InitializeComponent();

this.portfolioId = portfolioId;

this.userId = userId;

LoadPortfolioDetails();

}

private void makeReportButton\_Click(object sender, EventArgs e)

{

int days = (int)numericUpDown1.Value;

if (days <= 0)

{

MessageBox.Show("Please select a valid number of days.", "Invalid Input", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

DateTime endDate = DateTime.Now;

DateTime startDate = endDate.AddDays(-days);

var reportData = GetPortfolioValueHistory(portfolioId, startDate, endDate);

if (reportData.Count == 0)

{

MessageBox.Show("No data available for the selected period.", "No Data", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

string filePath = $"PortfolioReport\_{portfolioId}\_{DateTime.Now:yyyyMMddHHmmss}.xlsx";

GenerateExcelReport(filePath, reportData);

MessageBox.Show($"Report generated: {filePath}", "Report Generated", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

«Добавление нового пользователя в БД». Входные параметры: логин (String), пароль (String). Выходные параметры: Отсутствуют. Пример программного кода:

private void portfButton\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True");

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO [dbo].[userD] (login, pass, fio) VALUES (@Log, @Pass, @Fio)", con);

command.Parameters.Add("@Log", SqlDbType.VarChar).Value = textBox1.Text;

command.Parameters.Add("@Pass", SqlDbType.VarChar).Value = textBox2.Text;

command.Parameters.Add("@Fio", SqlDbType.VarChar).Value = textBox3.Text;

command.ExecuteNonQuery();

con.Close();

MessageBox.Show("Новый пользователь добавлен!");

RegForm frm = new RegForm(this.Location.X, this.Location.Y, this.Width, this.Height);

frm.Show();

this.Hide();

}

## Входные и выходные данные

Входные параметры — это данные, которые передаются в функцию или программу для обработки. Они могут быть различного типа, например, числовыми, строковыми, булевыми и т.д. Входные параметры используются для передачи информации, необходимой для выполнения определенной операции или расчета результата.

Выходные параметры — это данные, которые возвращаются функцией или программой после выполнения определенной операции или расчета результата. Они так же могут быть различного типа. Выходные параметры используются для передачи информации об результате выполнения функции или программы.

Входные и выходные параметры представлены на таблице 3.7.

Таблица 3.7 – Входные и выходные параметры функций ПС

|  |  |  |
| --- | --- | --- |
| Функция | Входные параметры | Выходные параметры |
| Добавление активов в портфель | вид актива, наименование, количество, цена. | Отсутствуют |
| Создание отчёта | количество дней, цена, дата изменения цены. | Отчёт в формате .xlsx  Результат представлен на рисунке Б.1. |
| Добавление нового пользователя в БД | имя, индивидуальный логин пользователя, пароль | Отсутствует |

## Функциональное тестирование

Тест-кейсы — это набор инструкций, которые описывают, как тестировать определенную функциональность программного обеспечения.

Он подробно описывает короткую последовательность действий, В тест-кейсе фиксируют подготовку к проверке, саму диагностику и ожидаемый результат, включая информацию о количестве проверок и нюансах. Они включают в себя шаги, которые должны быть выполнены тестировщиком, ожидаемые результаты и критерии для оценки успешности тестирования. Тест-кейсы позволяют убедиться в корректности и правильности работы программного продукта и выявить возможные ошибки или проблемы. Они также помогают повысить эффективность тестирования и сократить время, затрачиваемое тестировщиками на него.

Тест кейсы программного средства представлены на таблице 3.8.

Таблица 3.8 – Тест-кейсы

|  |  |  |  |
| --- | --- | --- | --- |
| Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| Добавление пользователя | Нажать «Нет аккаунта»  Ввести информацию о пользователе  Нажать «Зарегистрировать» | Добавление в базу данных нового пользователя | Добавление в базу данных нового пользователя. Результат представлен на рисунке B.1. |
| Вывод информации о портфелях | Ввести логин и пароль для входа в окно «Авторизация»  Нажать «Портфели»  Нажать «Детали портфеля» | Вывод информации о портфелях | Вывод информации о портфелях.  Результат представлен на рисунке B.2. |
| Расчет баланса портфеля | Ввести логин и пароль для входа в окно «Профиль»  Нажать «История транзакций» | Отображение текущего баланса и изменений | Отображение текущего баланса и изменений. Результат представлен на рисунке B.3. |
| Составление отчета | Ввести логин и пароль для входа в окно «Администратор»  Нажать «Отчёт»  Нажать «Отчёт в ворд» | Создание файла .docx с таблицей, содержащей отчёт | Создание файла .docx с таблицей, содержащей отчёт. Результат представлен на рисунке В.4. |

## Описание справочной системы

Справочная система разрабатывалась с помощью приложения Dr.Explain. Представляет из себя документ с одним разделом.

Раздел «1.» «Справочная система для приложения NorthVest» содержит в себе краткое пояснение ко всему функционалу программы, включая регистрацию и вход, информацию о пользователях, портфелях, балансу, ценным бумагам, отчетам, и другим.

Данную справочную систему можно вызвать нажатием на кнопку «?» в окне «Авторизация».

# Применение

## Назначение программного средства

Наименование разработанной ПС: «Программное средство для автоматизации учёта работы сотрудника инвестиционной компании».

Назначение ПС: облегчение процесса управления данными о сотрудниках и инвестиционных портфелях в инвестиционной компании. Приложение позволяет быстро и эффективно работать с данными пользователей, управлять инвестиционными портфелями и производить анализ финансовой информации.

Основные разработанные функции:

* авторизация пользователей;
* управление данными пользователей: возможность создания, просмотра, изменения и удаления данных о пользователях;
* управление портфелями: создание инвестиционных портфелей, добавление ценных бумаг, указание даты оформления;
* анализ данных: расчёт ежедневного, ежемесячного и общего изменения баланса портфеля, генерация графиков изменения баланса по различным ценным бумагам;
* поиск ценных бумаг: возможность поиска по различным критериям;
* генерация отчетов: отчеты о портфелях за выбранный промежуток времени, статистика по портфелям;
* экспорт данных: сохранение данных в файлы формата \*.xlsx.

Данное ПС позволяет повысить эффективность работы сотрудников инвестиционной компании, улучшить процесс управления портфелями и предоставить клиентам качественное обслуживание. Данный курсовой проект ограничивается разработкой ПС только для учёта и анализа данных в инвестиционной компании. Разработка других функций и возможностей, не связанных с указанными задачами, не является целью данного проекта. Также ограничения состоят на использовании языка программирования C# .

## Условия применения

Для функционирования программного средства для учёта работы сотрудников инвестиционной компании необходимо операционная система Windows 7 или выше, а также установленная СУБД Microsoft SQL Server 2012 или выше.

Для разработки программного средства использовался язык программирования C# и фреймворк .NET. Для удобства работы с базой данных была использована библиотека ADO.NET, которая позволяет осуществлять соединение с базой данных и выполнять запросы.

Процесс инсталляции программного средства состоит из нескольких шагов:

1. Установка операционной системы Windows 7 или выше.
2. Установка СУБД Microsoft SQL Server 2012 или выше.
3. Установка .NET Framework.
4. Скачивание и установка программного средства для учёта работы сотрудников инвестиционной компании «NorthVest».

После установки программного средства пользователь может начать работу с программой, управлять данными пользователей, создавать и анализировать инвестиционные портфели, просматривать отчеты и аналитические данные. Для работы данного программного средства требуется наличие на компьютере MS SQL Server, MS Excel и .NET Framework.

# Заключение

Это программное обеспечение было создано для автоматизации процессов учёта и управления данными в инвестиционной компании, что способствует ускорению рабочих процессов и минимизации ошибок, вызванных человеческим фактором. Разработка осуществлялась на платформе .NET Framework с использованием языка программирования C#. Все поставленные задачи были успешно выполнены.

Функционал программы включает в себя: регистрацию новых пользователей, управление инвестиционными портфелями, подготовку отчетности, автоматический расчёт изменения баланса портфелей и генерацию графиков изменений.

В ходе разработки возникли трудности с оптимизацией отображения данных, так как они формировались на основе данных из базы данных, что усложняло их использование.

Основным преимуществом программы является её способность координировать все ключевые операции инвестиционной компании, включая учет пользователей, управление портфелями и анализ данных, что даёт возможность эффективно управлять бизнесом и повышать его производительность.

Программа также обладает удобной системой хранения информации и позволяет составлять отчётность в файл(-ы) формата \*.xlsx.

Для дальнейшего улучшения программы предлагается исправление текущих ошибок, расширение функционала портфелей с возможностью выбора дополнительных параметров для каждой ценной бумаги, таких как тип, количество, что позволит клиентам лучше управлять своими инвестициями. Также полезной будет интеграция с программой лояльности, что позволит клиентам получать бонусы за использование программы, стимулируя тем самым повторное использование и повышая уровень их удовлетворенности.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2024. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2022. – 29 c.
3. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ‑Петербург, 2018. – 608 с.
4. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2022. – 816 с
5. Михнюк, Т.Ф. Охрана труда / Т.Ф. Михнюк. – Минск : ИВЦ Минфина, 2023. – 365 с
6. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2022. – 720 с.
7. Тепляков, С. Паттерны проектирования на платформе .NET / С.Тепляков. – СПб. : Питер, 2022. – 320 с.
8. Экономика предприятия. Практикум / Э. В. Крум [и др.] ; под ред. Э. В. Крум. – Минск : Издательство Гревцова, 2023. – 355 с.
9. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
10. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
11. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.

# Приложение А (обязательное)

**Текст программных модулей**

Файл RegForm

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace North\_Investments

{

public partial class RegForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

public RegForm(int locationX, int locationY, int width, int height)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

}

private void portfButton\_Click(object sender, EventArgs e)

{

SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True");

con.Open();

SqlCommand command = new SqlCommand("INSERT INTO [dbo].[userD] (login, pass, fio) VALUES (@Log, @Pass, @Fio)", con);

command.Parameters.Add("@Log", SqlDbType.VarChar).Value = textBox1.Text;

command.Parameters.Add("@Pass", SqlDbType.VarChar).Value = textBox2.Text;

command.Parameters.Add("@Fio", SqlDbType.VarChar).Value = textBox3.Text;

command.ExecuteNonQuery();

con.Close();

MessageBox.Show("Новый пользователь добавлен!");

RegForm frm = new RegForm(this.Location.X, this.Location.Y, this.Width, this.Height);

frm.Show();

this.Hide();

}

private void RegForm\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

}

Point lastPos;

private void RegForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void RegForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

private void button1\_Click(object sender, EventArgs e)

{

LoginForm loginForm = new LoginForm(this.Location.X, this.Location.Y, this.Width, this.Height);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

loginForm.Show();

this.Hide();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked)

{

textBox2.PasswordChar = '\0';

}

else

{

textBox2.PasswordChar = '\*';

}

}

}

}

Файл LoginForm

using North\_Investments.ExtraForms;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace North\_Investments

{

public partial class LoginForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

public LoginForm(int locationX, int locationY, int width, int height)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

}

private void LoginForm\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

Point lastPos;

private void LoginForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void LoginForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void portfButton\_Click(object sender, EventArgs e)

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand command = new SqlCommand("SELECT userId, fio, userBal FROM [dbo].[userD] WHERE login = @Login AND pass = @Pass", con);

command.Parameters.Add("@Login", SqlDbType.VarChar).Value = textBox1.Text;

command.Parameters.Add("@Pass", SqlDbType.VarChar).Value = textBox2.Text;

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

int userId = reader.GetInt32(0);

string fio = reader.GetString(1);

decimal userBalance = reader.GetDecimal(2);

DataSend.text = textBox1.Text;

DataSend.userId = userId;

DataSend.userBalance = userBalance;

ProfileForm profileForm = new ProfileForm(this.Location.X, this.Location.Y, this.Width, this.Height, userBalance);

profileForm.Show();

this.Hide();

}

else

{

MessageBox.Show("Неправильный логин или пароль");

}

}

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

RegForm regForm = new RegForm(this.Location.X, this.Location.Y, this.Width, this.Height);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

regForm.Show();

this.Hide();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked)

{

textBox2.PasswordChar = '\0';

}

else

{

textBox2.PasswordChar = '\*';

}

}

private void button1\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("\"D:\\NorthVest.chm\"");

}

}

}

Файл ProfileForm

using North\_Investments.ExtraForms;

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace North\_Investments

{

public partial class ProfileForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

private decimal userBalance;

public ProfileForm(int locationX, int locationY, int width, int height, decimal balance)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

userBalance = balance;

}

private void ProfileForm\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True");

string Sql = $"select fio from [dbo].[userD] where login = '{DataSend.text}'";

SqlCommand scmd = new SqlCommand(Sql, con);

con.Open();

SqlDataReader sur = scmd.ExecuteReader();

while (sur.Read())

{

string fio = sur["fio"].ToString();

label2.Text = fio;

}

con.Close();

userBalance = GetUserBalance();

balancelabel.Text = $"{userBalance:F2}";

LoadTransactionHistory();

}

private decimal GetUserBalance()

{

decimal balance = 0;

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("SELECT userBal FROM userD WHERE UserId = @UserId", con);

cmd.Parameters.AddWithValue("@UserId", DataSend.userId);

balance = (decimal)cmd.ExecuteScalar();

con.Close();

}

return balance;

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

Point lastPos;

private void HomePageForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void HomePageForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void homeButton\_Click(object sender, EventArgs e)

{

HomePageForm homePageForm = new HomePageForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

homePageForm.Show();

this.Close();

}

private void searchButton\_Click(object sender, EventArgs e)

{

SearchForm searchForm = new SearchForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

searchForm.Show();

this.Close();

}

private void portfButton\_Click(object sender, EventArgs e)

{

PortfoliosForm portfolios = new PortfoliosForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

portfolios.Show();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

LoginForm loginForm = new LoginForm(this.Location.X, this.Location.Y, this.Width, this.Height);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

loginForm.Show();

this.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

DialogResult MBresult = MessageBox.Show("Вы действительно хотите удалить текущего пользователя?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (MBresult == DialogResult.Yes)

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

try

{

con.Open();

string deleteSql = $"DELETE FROM [dbo].[userD] WHERE login = '{DataSend.text}'";

SqlCommand deleteCmd = new SqlCommand(deleteSql, con);

deleteCmd.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении пользователя: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

LoginForm loginForm = new LoginForm(this.Location.X, this.Location.Y, this.Width, this.Height);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

loginForm.Show();

this.Close();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Edit editForm = new Edit();

editForm.Show();

}

private void addBalButton\_Click(object sender, EventArgs e)

{

AddBalForm addBalForm = new AddBalForm();

addBalForm.Show();

}

private void LoadTransactionHistory()

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

string query = "SELECT TransactionType, Amount, TransactionDate FROM UserTransactions WHERE UserId = @UserId";

SqlCommand cmd = new SqlCommand(query, con);

cmd.Parameters.AddWithValue("@UserId", DataSend.userId);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

string transactionType = reader["TransactionType"].ToString();

decimal amount = (decimal)reader["Amount"];

DateTime transactionDate = (DateTime)reader["TransactionDate"];

Panel transactionPanel = new Panel

{

Size = new Size(100, 50),

BorderStyle = BorderStyle.None,

BackColor = Color.FromArgb(22, 22, 22),

Margin = new Padding(10)

};

Label typeLabel = new Label

{

Text = transactionType,

Location = new Point(10, 5),

Size = new Size(80, 15),

ForeColor = SystemColors.Control

};

Label amountLabel = new Label

{

Text = $"{amount:F2}",

Location = new Point(10, 20),

Size = new Size(80, 15),

ForeColor = SystemColors.Control

};

Label dateLabel = new Label

{

Text = transactionDate.ToShortDateString(),

Location = new Point(10, 35),

Size = new Size(80, 15),

ForeColor = SystemColors.Control

};

transactionPanel.Controls.Add(typeLabel);

transactionPanel.Controls.Add(amountLabel);

transactionPanel.Controls.Add(dateLabel);

flowLayoutPanel1.Controls.Add(transactionPanel);

}

reader.Close();

con.Close();

}

}

}

}

Файл HomePageForm

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace North\_Investments

{

public partial class HomePageForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

private int userId;

public HomePageForm(int locationX, int locationY, int width, int height, int userId)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

this.userId = userId;

LoadUserTransactions();

LoadAssetPriceHistory();

}

private void HomePageForm\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

}

private void LoadUserTransactions()

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("SELECT Amount, TransactionDate FROM UserTransactions WHERE UserId = @UserId ORDER BY TransactionDate", con);

cmd.Parameters.AddWithValue("@UserId", userId);

SqlDataReader reader = cmd.ExecuteReader();

chart1.Series.Clear();

Series series = new Series("Balance");

series.ChartType = SeriesChartType.Line;

chart1.Series.Add(series);

while (reader.Read())

{

decimal amount = (decimal)reader["Amount"];

DateTime transactionDate = (DateTime)reader["TransactionDate"];

series.Points.AddXY(transactionDate, amount);

}

con.Close();

}

}

private void LoadAssetPriceHistory()

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("SELECT AssetId FROM Assets WHERE Name = 'Ethereum'", con);

int ethereumAssetId = (int)cmd.ExecuteScalar();

SqlCommand cmdHistory = new SqlCommand("SELECT Price, RecordedDate FROM AssetPriceHistory WHERE AssetId = @AssetId ORDER BY RecordedDate", con);

cmdHistory.Parameters.AddWithValue("@AssetId", ethereumAssetId);

SqlDataReader reader = cmdHistory.ExecuteReader();

chart2.Series.Clear();

Series series = new Series("Ethereum Price");

series.ChartType = SeriesChartType.Line;

chart2.Series.Add(series);

while (reader.Read())

{

decimal price = (decimal)reader["Price"];

DateTime recordedDate = (DateTime)reader["RecordedDate"];

series.Points.AddXY(recordedDate, price);

}

con.Close();

}

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

Point lastPos;

private void HomePageForm\_MouseMove(object sender, MouseEventArgs e)

{

if(e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void HomePageForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void searchButton\_Click(object sender, EventArgs e)

{

SearchForm searchForm = new SearchForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

searchForm.Show();

this.Hide();

}

private void profileButton\_Click\_1(object sender, EventArgs e)

{

ProfileForm profileForm = new ProfileForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userBalance);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

profileForm.Show();

this.Hide();

}

private void portfButton\_Click(object sender, EventArgs e)

{

PortfoliosForm portfolios = new PortfoliosForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

portfolios.Show();

this.Hide();

}

}

}

Файл PortfoliosForm

using North\_Investments.ExtraForms;

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace North\_Investments

{

public partial class PortfoliosForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

private int userId;

public PortfoliosForm(int locationX, int locationY, int width, int height, int userId)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

this.userId = userId;

}

private void Portfolios\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

LoadPortfolios();

}

private void LoadPortfolios()

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("SELECT PortfolioId, PortfolioName FROM Portfolios WHERE UserId = @UserId", con);

cmd.Parameters.AddWithValue("@UserId", userId);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int portfolioId = (int)reader["PortfolioId"];

string portfolioName = reader["PortfolioName"].ToString();

AddPortfolioToPanel(portfolioId, portfolioName);

}

con.Close();

}

}

private void AddPortfolioToPanel(int portfolioId, string portfolioName)

{

Panel portfolioPanel = new Panel

{

Size = new Size(250, 100),

BorderStyle = BorderStyle.FixedSingle,

BackColor = Color.FromArgb(22, 22, 22),

Margin = new Padding(10)

};

Label nameLabel = new Label

{

Text = portfolioName,

TextAlign = ContentAlignment.MiddleCenter,

Dock = DockStyle.Top,

ForeColor = Color.White,

Height = 50

};

Button detailsButton = new Button

{

FlatStyle = FlatStyle.Flat,

Text = "Details",

Dock = DockStyle.Bottom,

ForeColor = Color.White,

BackColor = Color.FromArgb(66, 66, 66),

Height = 30

};

detailsButton.Click += (s, e) => OpenPortfolioDetails(portfolioId);

portfolioPanel.Controls.Add(nameLabel);

portfolioPanel.Controls.Add(detailsButton);

flowLayoutPanel1.Controls.Add(portfolioPanel);

}

private void OpenPortfolioDetails(int portfolioId)

{

PortfolioDetailsForm portfolioDetailsForm = new PortfolioDetailsForm(portfolioId, userId);

portfolioDetailsForm.Show();

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

Point lastPos;

private void HomePageForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void HomePageForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void homeButton\_Click(object sender, EventArgs e)

{

HomePageForm homePageForm = new HomePageForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

homePageForm.Show();

this.Hide();

}

private void searchButton\_Click(object sender, EventArgs e)

{

SearchForm searchForm = new SearchForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

searchForm.Show();

this.Hide();

}

private void profileButton\_Click(object sender, EventArgs e)

{

ProfileForm profileForm = new ProfileForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userBalance);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

profileForm.Show();

this.Hide();

}

}

}

Файл SearchForm

using North\_Investments.ExtraForms;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace North\_Investments

{

public partial class SearchForm : Form

{

private int previousLocationX;

private int previousLocationY;

private int previousWidth;

private int previousHeight;

private int userId;

private Dictionary<int, int> selectedAssets;

public SearchForm(int locationX, int locationY, int width, int height, int userId)

{

InitializeComponent();

previousLocationX = locationX;

previousLocationY = locationY;

previousWidth = width;

previousHeight = height;

this.userId = userId;

selectedAssets = new Dictionary<int, int>();

}

private void SearchForm\_Load(object sender, EventArgs e)

{

this.Location = new Point(previousLocationX, previousLocationY);

this.Width = previousWidth;

this.Height = previousHeight;

LoadAssets();

}

private void LoadAssets()

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

LoadAssetsFromDatabase(con, "SELECT AssetId, Name, Price FROM Assets WHERE Type = 'Security'", flowLayoutPanel1);

LoadAssetsFromDatabase(con, "SELECT AssetId, Name, Price FROM Assets WHERE Type = 'Stock'", flowLayoutPanel2);

LoadAssetsFromDatabase(con, "SELECT AssetId, Name, Price FROM Assets WHERE Type = 'Cryptocurrency'", flowLayoutPanel3);

con.Close();

}

}

private decimal GetUserBalance()

{

decimal balance = 0;

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("SELECT userBal FROM userD WHERE UserId = @UserId", con);

cmd.Parameters.AddWithValue("@UserId", userId);

balance = (decimal)cmd.ExecuteScalar();

con.Close();

}

return balance;

}

private void UpdateUserBalance(decimal newBalance)

{

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand("UPDATE userD SET userBal = @userBal WHERE UserId = @UserId", con);

cmd.Parameters.AddWithValue("@userBal", newBalance);

cmd.Parameters.AddWithValue("@UserId", userId);

cmd.ExecuteNonQuery();

con.Close();

}

}

private void LoadAssetsFromDatabase(SqlConnection con, string query, FlowLayoutPanel panel)

{

SqlCommand cmd = new SqlCommand(query, con);

SqlDataReader reader = cmd.ExecuteReader();

while (reader.Read())

{

int assetId = (int)reader["AssetId"];

string assetName = reader["Name"].ToString();

decimal price = (decimal)reader["Price"];

AddAssetToPanel(assetId, assetName, price, panel);

}

reader.Close();

}

private void AddAssetToPanel(int assetId, string assetName, decimal price, FlowLayoutPanel panel)

{

Panel assetPanel = new Panel

{

Size = new Size(80, 80),

BorderStyle = BorderStyle.FixedSingle,

BackColor = Color.FromArgb(22, 22, 22),

Margin = new Padding(10),

Tag = assetId

};

Label nameLabel = new Label

{

Text = $"{assetName}\n${price}",

TextAlign = ContentAlignment.MiddleCenter,

Dock = DockStyle.Fill,

ForeColor = SystemColors.Control

};

assetPanel.Controls.Add(nameLabel);

assetPanel.Click += (s, e) => ToggleAssetSelection(assetPanel, assetId);

nameLabel.Click += (s, e) => ToggleAssetSelection(assetPanel, assetId);

panel.Controls.Add(assetPanel);

}

private int ShowAssetAmountFormCentered()

{

using (AssetAmountForm amountForm = new AssetAmountForm())

{

amountForm.StartPosition = FormStartPosition.CenterParent;

if (amountForm.ShowDialog() == DialogResult.OK)

{

return (int)amountForm.Amount;

}

else

{

return 0;

}

}

}

private void ToggleAssetSelection(Panel assetPanel, int assetId)

{

if (selectedAssets.ContainsKey(assetId))

{

selectedAssets.Remove(assetId);

assetPanel.BackColor = Color.FromArgb(22, 22, 22);

}

else

{

int amount = ShowAssetAmountFormCentered();

if (amount > 0)

{

selectedAssets.Add(assetId, amount);

assetPanel.BackColor = Color.Gray;

}

}

}

private string GetPortfolioName()

{

using (SetPortfolioName portfolioNameForm = new SetPortfolioName())

{

portfolioNameForm.StartPosition = FormStartPosition.CenterParent;

if (portfolioNameForm.ShowDialog() == DialogResult.OK)

{

return portfolioNameForm.PortfolioName;

}

else

{

return null;

}

}

}

private void CreatePortfButton3\_Click(object sender, EventArgs e)

{

if (selectedAssets.Count == 0)

{

MessageBox.Show("Please select at least one asset for the portfolio.");

return;

}

string portfolioName = GetPortfolioName();

if (string.IsNullOrEmpty(portfolioName))

{

MessageBox.Show("Please provide a valid name for the portfolio.");

return;

}

decimal totalCost = 0;

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

using (SqlTransaction transaction = con.BeginTransaction())

{

try

{

foreach (var asset in selectedAssets)

{

SqlCommand priceCmd = new SqlCommand("SELECT Price FROM Assets WHERE AssetId = @AssetId", con, transaction);

priceCmd.Parameters.AddWithValue("@AssetId", asset.Key);

decimal price = (decimal)priceCmd.ExecuteScalar();

totalCost += price \* asset.Value;

}

decimal currentBalance = GetUserBalance();

if (currentBalance < totalCost)

{

MessageBox.Show("Insufficient balance to purchase selected assets.");

transaction.Rollback();

return;

}

SqlCommand cmd = new SqlCommand("INSERT INTO Portfolios (UserId, PortfolioName) OUTPUT INSERTED.PortfolioId VALUES (@UserId, @PortfolioName)", con, transaction);

cmd.Parameters.AddWithValue("@UserId", userId);

cmd.Parameters.AddWithValue("@PortfolioName", portfolioName);

int portfolioId = (int)cmd.ExecuteScalar();

foreach (var asset in selectedAssets)

{

SqlCommand insertCmd = new SqlCommand("INSERT INTO PortfolioDetails (PortfolioId, AssetId, Quantity) VALUES (@PortfolioId, @AssetId, @Quantity)", con, transaction);

insertCmd.Parameters.AddWithValue("@PortfolioId", portfolioId);

insertCmd.Parameters.AddWithValue("@AssetId", asset.Key);

insertCmd.Parameters.AddWithValue("@Quantity", asset.Value);

insertCmd.ExecuteNonQuery();

}

decimal newBalance = currentBalance - totalCost;

UpdateUserBalance(newBalance);

string insertTransactionSql = "INSERT INTO UserTransactions (UserId, TransactionType, Amount, TransactionDate) VALUES (@UserId, @TransactionType, @Amount, @TransactionDate)";

SqlCommand insertTransactionCmd = new SqlCommand(insertTransactionSql, con, transaction);

insertTransactionCmd.Parameters.AddWithValue("@UserId", userId);

insertTransactionCmd.Parameters.AddWithValue("@TransactionType", "Трата");

insertTransactionCmd.Parameters.AddWithValue("@Amount", totalCost);

insertTransactionCmd.Parameters.AddWithValue("@TransactionDate", DateTime.Now);

insertTransactionCmd.ExecuteNonQuery();

transaction.Commit();

MessageBox.Show("Portfolio created successfully.");

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show("Ошибка при создании портфеля: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void closeAppLabel\_Click(object sender, EventArgs e)

{

Program.CloseAllForms();

}

private void closeAppLabel\_MouseEnter(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.Red;

}

private void closeAppLabel\_MouseLeave(object sender, EventArgs e)

{

closeAppLabel.ForeColor = Color.White;

}

Point lastPos;

private void HomePageForm\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPos.X;

this.Top += e.Y - lastPos.Y;

}

}

private void HomePageForm\_MouseDown(object sender, MouseEventArgs e)

{

lastPos = new Point(e.X, e.Y);

}

private void homeButton\_Click(object sender, EventArgs e)

{

HomePageForm homePageForm = new HomePageForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

homePageForm.Show();

this.Hide();

}

private void portfButton\_Click(object sender, EventArgs e)

{

PortfoliosForm portfolios = new PortfoliosForm(this.Location.X, this.Location.Y, this.Width, this.Height, DataSend.userId);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

portfolios.Show();

this.Hide();

}

private void profileButton\_Click(object sender, EventArgs e)

{

decimal currentBalance = GetUserBalance();

ProfileForm profileForm = new ProfileForm(this.Location.X, this.Location.Y, this.Width, this.Height, currentBalance);

previousLocationX = this.Location.X;

previousLocationY = this.Location.Y;

previousWidth = this.Width;

previousHeight = this.Height;

profileForm.Show();

this.Hide();

}

private void PerformSearch(string name, string minPrice, string maxPrice)

{

string query = "SELECT AssetId, Name, Price, Type FROM Assets WHERE 1=1";

if (!string.IsNullOrEmpty(name))

{

query += " AND Name LIKE @Name";

}

if (decimal.TryParse(minPrice, out decimal min))

{

query += " AND Price >= @MinPrice";

}

if (decimal.TryParse(maxPrice, out decimal max))

{

query += " AND Price <= @MaxPrice";

}

using (SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-S1LVDJS\SQLEXPRESS;Initial Catalog=NorthVest;Integrated Security=True"))

{

con.Open();

SqlCommand cmd = new SqlCommand(query, con);

if (!string.IsNullOrEmpty(name))

{

cmd.Parameters.AddWithValue("@Name", "%" + name + "%");

}

if (decimal.TryParse(minPrice, out min))

{

cmd.Parameters.AddWithValue("@MinPrice", min);

}

if (decimal.TryParse(maxPrice, out max))

{

cmd.Parameters.AddWithValue("@MaxPrice", max);

}

SqlDataReader reader = cmd.ExecuteReader();

flowLayoutPanel1.Controls.Clear();

flowLayoutPanel2.Controls.Clear();

flowLayoutPanel3.Controls.Clear();

while (reader.Read())

{

int assetId = (int)reader["AssetId"];

string assetName = reader["Name"].ToString();

decimal price = (decimal)reader["Price"];

string assetType = reader["Type"].ToString();

switch (assetType)

{

case "Security":

AddAssetToPanel(assetId, assetName, price, flowLayoutPanel1);

break;

case "Stock":

AddAssetToPanel(assetId, assetName, price, flowLayoutPanel2);

break;

case "Cryptocurrency":

AddAssetToPanel(assetId, assetName, price, flowLayoutPanel3);

break;

}

}

reader.Close();

con.Close();

}

}

private void searchButton2\_Click(object sender, EventArgs e)

{

string name = nameTextBox.Text;

string minPrice = minPriceTextBox.Text;

string maxPrice = maxPriceTextBox.Text;

PerformSearch(name, minPrice, maxPrice);

}

}

}

# Приложение Б (справочное) Формы входных и выходных документов

Входным документом является база данных SQL, содержащая информацию о продукции и её цене.

Выходными параметрами является документ расширения .xlsx, содержащий дату и соответствующую цену портфеля в это время.

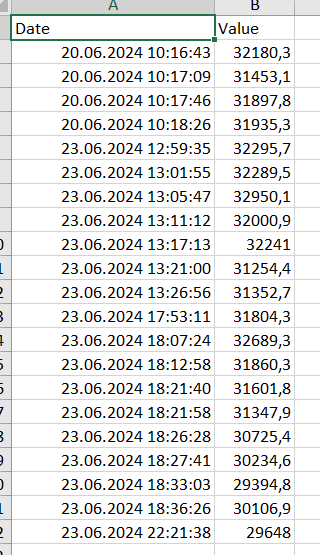


Рисунок Б.1 – Отчет в Exel

# Приложение В (обязательное) Результаты работы программы

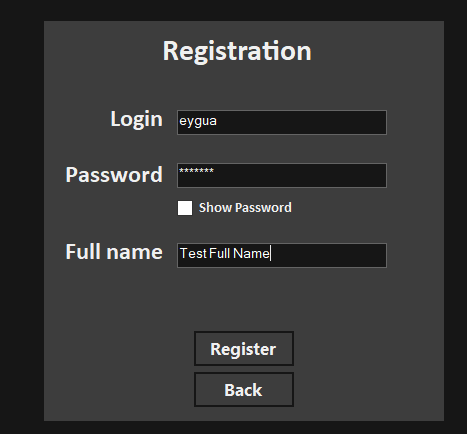


Рисунок В.1 – Добавление пользователя в БД

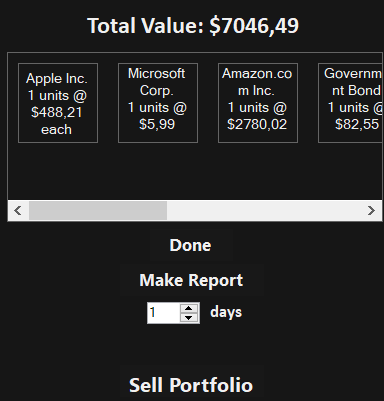


Рисунок В.2 – Вывод деталей портфеля

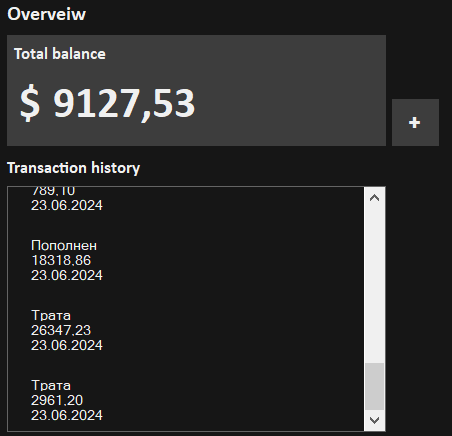


Рисунок В.2 – Отображение текущего баланса и изменений.

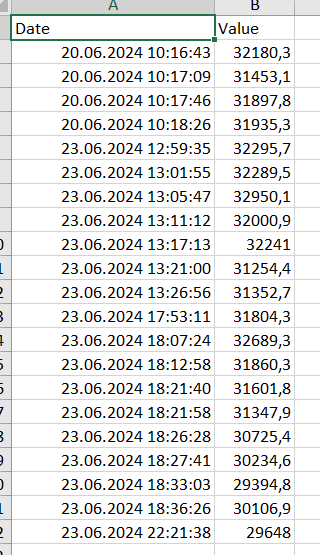


Рисунок В.4 – Составление отчета

КП Т.193021.401 ГЧ

КБП

У

Т. Контр.

Разраб. Разраб.

Найдун Р.М

Провер. Провер.

Елинская Д.О

Реценз.

Утверд.

Н. Контр.

Диаграмма вариантов использования

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 4

Инв.№подл.

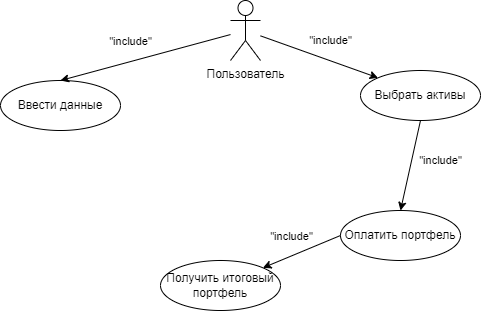
Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.193021.401 ГЧ Т.ХХХХХХ.401 ГЧ



Программное средство для Автоматицазии учёта работы сотрудника инвестиционной компании

Программное средство для управления заказами в пиццерии «ПИЦЦА У КИБИПЦА»

КП Т.193021.401 ГЧ

КБП

У

Т. Контр.

Разраб. Разраб.

Найдун Р.М

Провер. Провер.

Елинская Д.В

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 4

Инв.№подл.

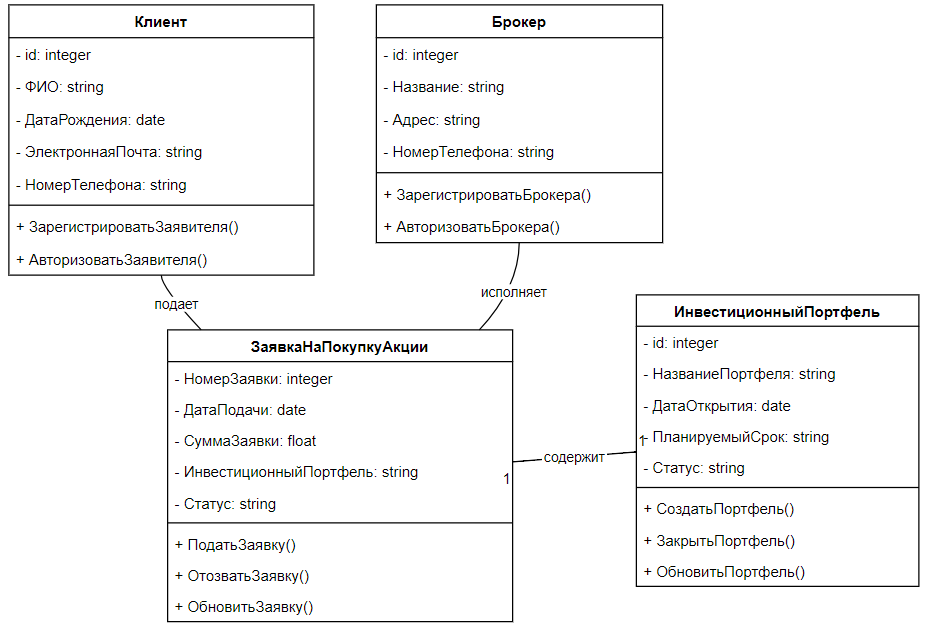
Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.193021.401 ГЧ Т.ХХХХХХ.401 ГЧ



Программное средство для Автоматицазии учёта работы сотрудника инвестиционной компании

Программное средство для управления заказами в пиццерии «ПИЦЦА У КИБИПЦА»

Диаграмма Классов

КП Т.193021.401 ГЧ

КБП

У

Т. Контр.

Разраб. Разраб.

Найдун Р.М

Провер. Провер.

Елинская Д.О

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 3

Листов 4

Инв.№подл.

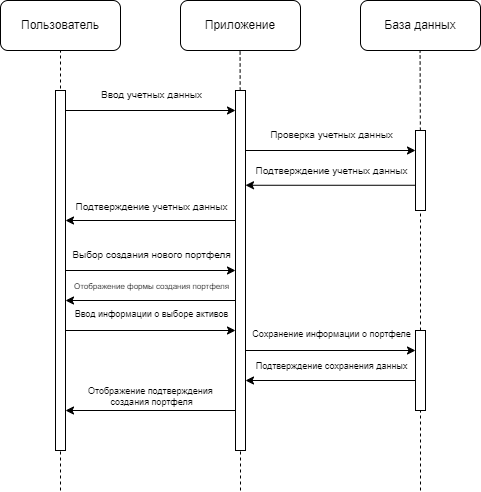
Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.193021.401 ГЧ Т.ХХХХХХ.401 ГЧ



Программное средство для Автоматицазии учёта работы сотрудника инвестиционной компании

Диаграмма последовательности

Программное средство для управления заказами в пиццерии «ПИЦЦА У КИБИПЦА»

КБП

У

Т. Контр.

Разраб. Разраб.

Найдун Р.М

Провер. Провер.

Елинская Д.В

Реценз.

Утверд.

Н. Контр.

Диаграмма деятельности

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 4

Листов 4

Инв.№подл.

Подп. и дата

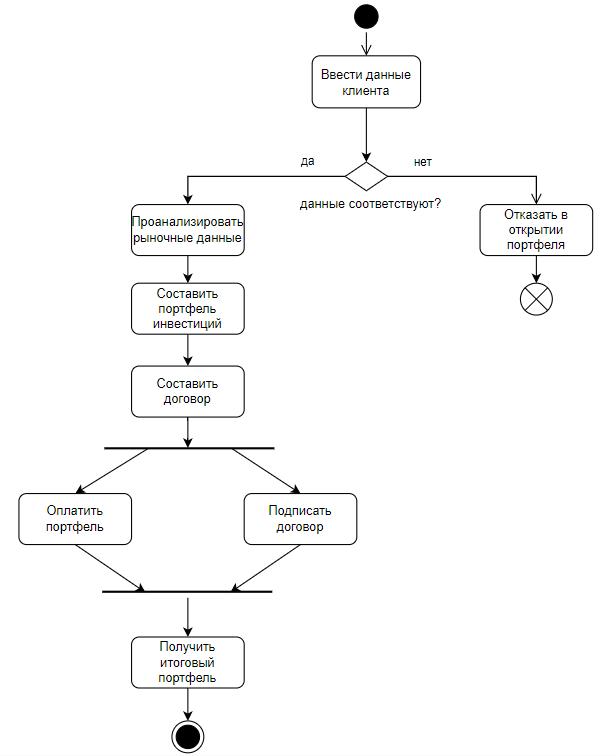
Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.193021.401 ГЧ Т.ХХХХХХ.401 ГЧ

КП Т.193021.401 ГЧ



Сохранить данные

Оформить портфель

Программное средство для Автоматицазии учёта работы сотрудника инвестиционной компании

Программное средство для управления заказами в пиццерии «ПИЦЦА У КИБИПЦА»

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема КП «Разработка программного средства для автоматизации учёта работы сотрудника инвестиционной компании"

Обозначение КПТ.193021.401

Разработчик \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), ПЗ\_НайдунТ193.docх |  |  |
| ГЧ, Диаграмма вариантов использования.docx |  |  |
| ГЧ, ДиаграммаКлассов.docx |  |  |
| ГЧ, Диаграмма последовательности.docx |  |  |
| ГЧ, Диаграмма деятельности.docx |  |  |
| Папка с проектом «North Investments» |  |  |
| Установочный пакет программного средства «North Investments.exe» |  |  |
| Файл базы данных «North Investments.mdf» |  |  |
| Тип носителя: оптический диск |  |  |

**Этикетка**

для курсового проекта

**Курсовой проект**

Тема «Разработка программного средства для автоматизации учёта работы сотрудника инвестиционной компании»

КП Т.193021.401

Разработан

Утвержден

Разработчик Найдун Р.М.

Руководитель: Елинская Д.О.

Технические средства: Компьютер, мышь, клавиатура.

Программные средства: MS SQL Server, Net Framework, MS Word.

**Состав документа:**

Пояснительная записка – ПЗ\_НайдунТ193.docх

Графическая часть – Диаграмма вариантов использования.docx, ДиаграммаКлассов.docx, Диаграмма последовательности.docx, Диаграмма деятельности.docx

Папка с проектом – North Investments

Установочный пакет программного средства – North Investments.exe

Файл базы данных – North Investments.mdf