**Министерство профессионального образования и занятости населения Приморского края**

**краевое государственное автономное профессиональное**

**образовательное учреждение**

**«ПРИМОРСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

**(КГА ПОУ «ППК»)**

Отделение Экономики и сервиса

Специальность *09.02.07*

«Информационные системы и программирование»

Работа допущена к защите:

Завотделением \_\_\_\_\_\_\_ А.А. Савина

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**КУРСОВОЙ ПРОЕКТ**

**по теме:**

**АВТОМАТИЗАЦИЯ УЧЕТА ДОХОДОВ И РАСХОДОВ**

Дисциплина: Основы алгоритмизации и программирования

|  |  |
| --- | --- |
| **Выполнил**  студент группы 1125 | **Руководитель**  преподаватель |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Е. Приказюк | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Пушкин |
| Оценка, полученная на защите КП, | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Владивосток 2023**

**Министерство профессионального образования и занятости населения Приморского края**

**краевое государственное автономное профессиональное**

**образовательное учреждение**

**«ПРИМОРСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

**(КГА ПОУ «ППК»)**

**Отделение Экономики и сервиса**

**З А Д А Н И Е**

на курсовой проект

Заведующий отделением \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Савина

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

|  |
| --- |
| *Студенту Приказюку Ивану Евгеньевичу 1125 группы* |

(фамилия, имя, отчество)

на тему «Автоматизация учета доходов и расходов»

|  |
| --- |
|  |
|  |

Вопросы, подлежащие разработке (исследованию):

|  |
| --- |
| 1. Изучение предметной области. Выделение объектов и связей между ними. |
| 1. Разработка схемы базы данных |
| 1. Построение диаграммы объектов. |
| 1. Написание технического задания на разработку системы |
| 1. Разработка функциональной части приложения |
| 1. Разработка пользовательского интерфейса приложения |
|  |

Основные источники информации и прочее, используемые для разработки темы

|  |
| --- |
| 1. Гаурав Арораа, Джефри Чилберто – Паттерны проектирования для C# и платформы .Net Core, 2021 |
| 1. Эндрю Лок – Asp.Net Core в действии, 2021 |
| 1. Прайс Марк – C# 10 и .Net 6. Современная кросс-платформенная разработка, 2023 |
|  |

Срок предоставления работы «01» апреля 2023 г.

Дата выдачи задания «17» мая 2023 г.

Руководитель КП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Пушкин

(подпись) (и.о. фамилия)

Задание получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Е. Приказюк

(подпись) (и.о. фам

СОДЕРЖАНИЕ

[Термины и определения 4](#_Toc135353309)

[Введение 5](#_Toc135353310)

[1 Проектирование разрабатываемой системы 6](#_Toc135353311)

[1.1 Выявление основных сущностей и их взаимосвязей 6](#_Toc135353312)

[1.2 Формирование требований к разрабатываемой системе 7](#_Toc135353313)

[1.3 Разработка диаграмм деятельности 11](#_Toc135353314)

[1.4 Разработка схемы базы данных 14](#_Toc135353315)

[1.5 Используемые технологии 16](#_Toc135353316)

[2 Описание разработанной системы 18](#_Toc135353317)

[2.1 Описание интерфейса пользователя 18](#_Toc135353318)

[2.2 Пользовательские сценарии 19](#_Toc135353319)

[Заключение 24](#_Toc135353320)

[Список используемых источников 25](#_Toc135353321)

[Приложение А – Страница регистрации 26](#_Toc135353322)

[Приложение Б – Страница авторизации 27](#_Toc135353323)

[Приложение В – Главная страница WPF приложения 28](#_Toc135353324)

[Приложение Г – Окно статистики 29](#_Toc135353325)

Термины и определения

В настоящем курсовом проекте применяются следующие термины с соответствующими определениями:

1. **База данных** (**БД**) — это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области.
2. **Программа** — это описание на формальном языке, «понятном» компьютеру, последовательности действий, которые необходимо выполнить над данными для решения поставленной задачи.
3. **WPF** (Windows Presentation Foundation) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем.
4. **WinForms** (Windows Forms) — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio.
5. **Windows** — группа семейств коммерческих проприетарных операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса.
6. **Visual Studio** — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений.
7. **Скриншот** (от англ. screenshot) — это снимок экрана вашего компьютера или мобильного устройства.

Введение

В наше время управление персональными финансами становится всё сложнее, это связано с постоянными изменениями в ценообразовании магазинов и прочих предприятий. Чтобы решить проблему с подсчетом финансов, всё больше людей и компаний начинают использовать автоматизированные системы учета доходов и расходов, создавая отчеты и анализируя данные, благодаря которым становится легче принять решения в распоряжении своими финансами.

Целью данного курсового проекта является разработка WPF приложения для автоматизации учета доходов и расходов за определенные промежутки времени. Из этого вытекают следующие задачи:

1. Выявить основные сущности и их взаимосвязи;
2. Определить основные алгоритмы, разрабатываемого приложения;
3. Разработать приложение, помогающее автоматизировать учет доходов и расходов;
4. Протестировать разработанное приложение.

**1 Проектирование разрабатываемой системы**

* 1. **Выявление основных сущностей и их взаимосвязей**

В приложении есть основные объекты:

1) Аккаунт пользователя;

2) Статистика пользователя.

В приложении запланировано, что у аккаунта пользователя есть:

1. ФИО
2. Логин
3. Пароль
4. Статистика

В статистике в свою очередь хранятся данные:

1. Доходов;
2. Расходов;
3. Сумма дохода/расхода;
4. Баланс.

В соответствии с рисунком 1, пользователь должен авторизоваться в приложении, прежде чем начать использование. Если пользователь неавторизованный, то ему не будет доступен полный функционал приложения. Авторизованный пользователь уже может пользоваться всеми функциями приложения:

1. Выбор категории используемых средств (Доход/Расход);
2. Выбор категории получения/списания средств;
3. Добавление записи о пополнении/списании;
4. Просмотр статистики;

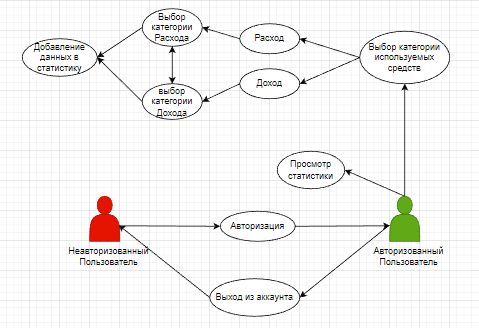


Рисунок 1

В зависимости от выбранной категории используемых средств, статистика будет автоматически рассчитывать данные о пополнении/списании средств поступающих на счет пользователя. А также статистика будет в режиме реального времени синхронизироваться с активным аккаунтом пользователя.

1.2 Формирование требований к разрабатываемой системе

Для начала необходимо разобраться, что такое функциональные требования.

Функциональные требования — это постановка задачи разработчику. Все, что не указано в требованиях, делается на усмотрение разработчика, что часто расходится с представлением продукт-менеджера об ожидаемом результате. Поэтому требования должны содержать ответы на все возможные вопросы по задаче.

Функциональные требования:

* User story (Пользовательские истории) — показывает, чего вы ожидаете от команды разработки;
* Use cases — показывают сценарии использования;
* Wireframes — средство визуализации своей идеи.

Пользовательская история, или User Story — это описание функций продукта простым языком, составленное с точки зрения пользователя. Она помогает понять, какую пользу клиенту принесет функционал приложения, еще на этапе аналитики проекта. Пользовательские истории служат неким контекстом для разработчиков: те понимают, чего хочет от продукта конечный пользователь, и работают более целенаправленно. Эти истории состоят из нескольких предложений и не углубляются в детали: они отражают суть и фокусируются на главном.

User Story (Пользовательские истории) — помогают сосредоточиться на потребностях пользователя. Например: как пользователь будет использовать приложение? Чего пользователь ждёт от продукта? Как пользователь поведёт себя в той или иной ситуации? Таким образом, ответы на эти вопросы помогут разработчикам продукта решать настоящие проблемы клиентов. Вот несколько главных задач, для которых необходимо использовать User Storys:

* Организовать работу. Когда проект разбит на части, связанные с пользовательскими историями, каждая из них представляет собой цельную и понятную задачу. Так разработчик может сфокусироваться на каждой из них и получать измеримый результат;
* Cохранить фокус на пользователе. Конечно, разработка включает себя десятки сложных задач, связанных с техническими, финансовыми и другими вопросами. Однако User Storys — это постоянное напоминание разработчику о тех, для кого этот продукт создается, и направляют его работу в нужное русло.

Хоть каждая пользовательская история и уникальна, но у них всех имеются стандартные элементы создания пользовательской истории, которые позволяют лучше всего понять пользователя приложения. Эти элементы включают в себя:

1. Заголовок – краткое описание истории;
2. Пользователь как – роль;
3. Пользователю нужно – функционал;
4. С целью… – польза.

Необходимо создать приложение, которое позволит пользователю упростить ведение учёта доходов и расходов. Также приложение должно предоставлять пользователю статистику полученных и списанных со счета средств. Необходима возможность использования пользователем статистики для грамотного подсчета средств и их распределения.

На основе данных требований, можно составить следующие пользовательские истории для разрабатываемого приложения:

* Пользователю требуется возможность выбирать категорию и сумму средств;
* Пользователю требуется возможность просматривать добавленные средства и их категорию;
* Пользователю нужно, чтобы в приложении к средствам была прикреплена дата их получения или списания.

Use case — это сценарий взаимодействия пользователя или пользователей с программным продуктом для достижения конкретной цели.

Use case содержит следующие сведения:

* Кто использует сайт или приложение;
* Что пользователю нужно сделать;
* Цель пользователя;
* Шаги, которые делает пользователь, чтобы совершить определенное действие;
* Описание того, как приложение реагируют на действия пользователя.

Use case не содержит детали реализации, а также описания пользовательского интерфейса или экранов.

В общем, в Use case описывается, не каким образом программа делает что-либо, а что именно она делает. Именно этого подхода и нужно придерживаться, создавая Use case.

В отличие от пользовательской истории, которая излагается от имени какого-то конкретного пользователя, в Use case может быть описано взаимодействие (с определенной целью) нескольких участников. Например:

* покупка товара в магазине (Покупатель — Продавец);
* отправка письма по электронной почте (Отправитель — Почтовый клиент);
* запрос страницы браузером (браузер — веб–сервер).

Use case может содержать следующие элементы. Их количество зависит от сценария:

* Актёр (actor) — тот, кто использует систему. Если взять за пример онлайн–магазин, там может быть несколько актёров: покупатели, продавцы, компании, занимающиеся доставкой, компании, проводящие платежи;
* Стейкхолдер (stakeholder) — тот, кто заинтересован в определенном поведении системы. Зачастую это не конечный пользователь, а кто-то, получающий выгоду от функционирования системы. В случае с онлайн–магазином это может быть партнер — платежная платформа;
* Первичное действующее лицо (primary actor) — человек или система, чьи цели достигаются при помощи нашего продукта. В онлайн–магазине это может быть основной дистрибьютор, чьи товары продаются на этой онлайн–платформе;
* Предусловия и постусловия — что должно быть в наличии или должно произойти до и после запуска сценария использования;
* Триггеры — события, запускающие Use case;
* Успешный сценарий — Use case, при котором все идет по плану, без ошибок и неожиданностей;
* Альтернативные пути — вариации основного успешного сценария на случай, если что-то пойдет не так на уровне системы.

Use case не обеспечивают полноту всех функциональных требований, если в систему должна быть заложена сложная бизнес–логика, то есть обработка информации в системе зависит не только и не столько от действий пользователей, сколько от внутренних правил взаимодействия объектов.

У разрабатываемого приложения есть минимальные системные требования:

1. Процессор с тактовой частотой не менее 2 ГГц;
2. Не менее 4 Гб оперативной памяти;
3. Не менее 256 Мб видеопамяти;
4. Не менее 255 Мб свободного места на диске.

Работать приложение сможет на таких операционных системах:

1. Windows 7 SP1 (32- и 64-разрядные версии);
2. Windows 8.1 (32-разрядная версия, 64-разрядная версия и ARM);
3. Windows 10 начиная с юбилейного обновления «Anniversary Update»  
   (версия 1607) (32- и 64-разрядные версии);
4. Windows 11 (64-разрядная система);

1.3 Разработка диаграмм деятельности

Диаграмма Use case является первой из диаграмм, разрабатываемых при моделировании предметной области системы или программного обеспечения. Является базой при разработке спецификации функциональных требований. И имеют основополагающие значения с точки зрения полноты и корректности дальнейшего моделирования проектируемой системы. Прецедент описывает типичное взаимодействие между пользователем и системой. Прецедент это вариант использования. Прецедент это последовательность действий выполняемых системой в ответ на событие, инициируемое к некоторым внешним объектам (user, другие системы или другие устройства). Каждый вариант использования связан с некоторой целью. В зависимости от цели выполнения различают следующие варианты использования:

1. Основные — обеспечивают требуемую функциональность разрабатываемого ПО;
2. Вспомогательные — обеспечивают выполнение необходимых настроек системы и её обслуживание;
3. Дополнительные — обеспечивает дополнительные удобства для пользователя.

У проекта имеется некоторое количество диаграмм деятельности по прецедентам:

1. Авторизация. Пользователю в самом начале использования приложения необходимо подключиться к соответствующей базе данных (авторизоваться). В соответствии с рисунком 2, открывая приложение, пользователь сразу попадает на главный экран приложения, после этого пользователь должен перейти в окно настроек, где и произведёт авторизацию. Для авторизации требуется ввести адрес сервера, название базы данных, логин и пароль. Если данные введены неверно или сервер выключен, то пользователь должен ввести верные данные или включить сервер. Также пользователь может сохранить введенные им данные, чтобы каждый раз их не вводить;

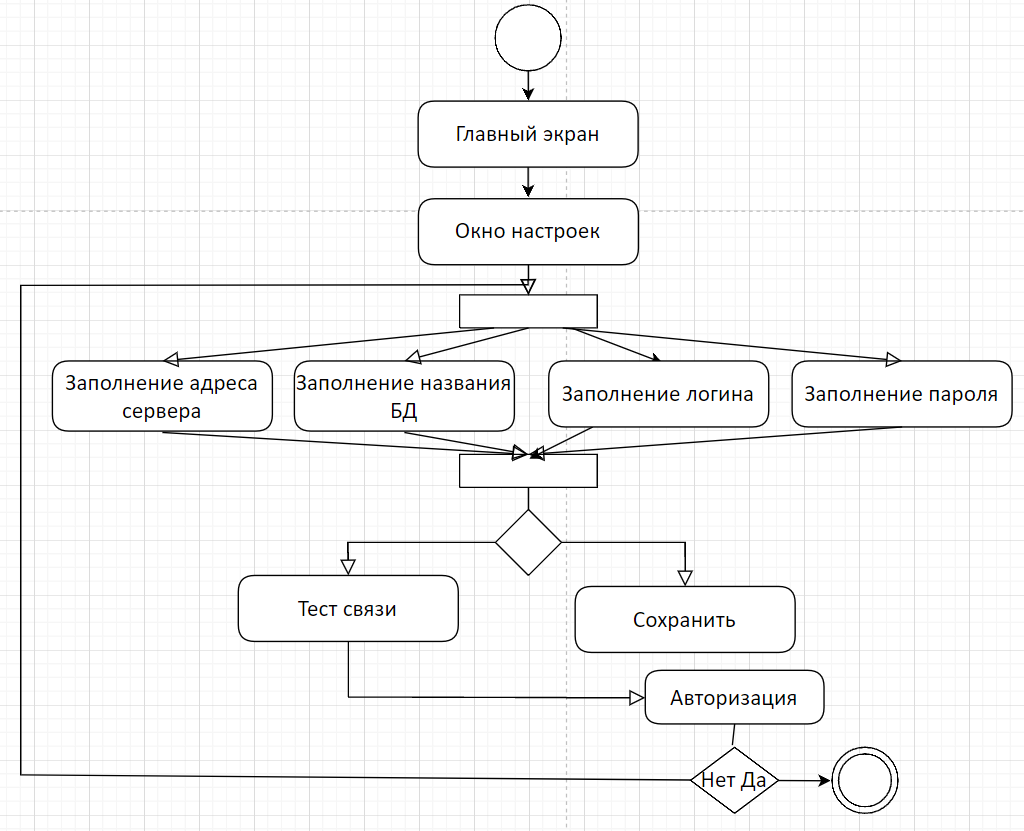


Рисунок 2 Авторизация

1. Добавление финансов. Пользователь может сохранить категорию финансов в приложении. В соответствии с Приложением Б, пользователь после открытия приложения попадает на главный экран, откуда уже может создать запись о поступлении или списании средств. Пользователь должен выбрать категорию и ввести сумму средств. После этого он может сохранить введённые данные;
2. Просмотр статистики. В соответствии с Приложением В, открыв приложение, пользователь должен перейти на страницу «Статистика», чтобы увидеть список всех поступлений или списаний и там же можно посмотреть их дату;
3. Весь функционал приложения. В соответствии с Приложением А, при открытии приложения, пользователь попадает на экран, где он совершит авторизацию или регистрацию нового аккаунта. Далее пользователь может добавить запись о поступлении или списании средств, указав перед этим их категорию, важно учесть что первой транзакцией нового пользователя не может стать трата средств. Перейдя на страницу статистики пользователь может увидеть список всех транзакций их категорию, тип, дату и текущий баланс, а также, автоматически созданные исходя из всех записей итоги за день, месяц, год и всё время эксплуатации приложения. В случае необходимости в смене аккаунта, пользователь может нажать кнопку «Выйти», что приведет к выходу из аккаунта.

1.4 Разработка схемы базы данных

Для разрабатываемого приложения есть база данных. Без базы данных приложение можно сказать бессмысленно. Построим ERD Диаграмму для наглядности. ERD(сущность-связь) диаграмма используется в основном для проектирования схемы БД. На диаграмме отображаются сущности, их атрибуты и связи между этими сущностями.

Шаги построения ERD:

1. На естественном языке составляется как можно более полное и непротиворечивое описание разрабатываемого приложения;
2. В тексте подчёркиваются все существительные;
3. Глаголы подчёркиваются двумя чертами;
4. Существительные выписываются отдельно и без повторов;
5. На основании глаголов проводятся линии между связанными существительными;
6. Изучение существительных без связей;
7. Расставляются типы связей (один к одному, многие к одному, многие ко многим);
8. Существительные, связанные связью один к одному, объединяются в одну сущность;
9. Связи многие ко многим преобразуются в кросс–сущность;
10. Добавляем типы и ключевые поля;

В соответствии с рисунком 3, в базе данных есть 6 таблиц:

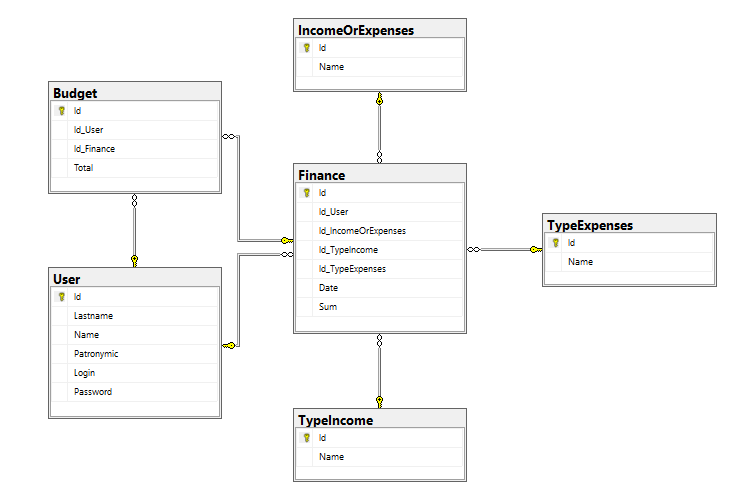


Рисунок 3

1. Finance — хранит в себе id, пользователя (колонка «Id\_User»), тип транзакции (колонка «Id\_IncomeOrExpenses»), тип дохода (колонка «Id\_TypeIncome»), тип расхода (колонка «Id\_TypeExpenses»), дату (колонка «Date»), сумму транзакции (колонка «Sum»);
2. Budget — хранит в себе id, пользователя (колонка «Id\_User»), транзакцию (колонка «Id\_Finance») и баланс (колонка «Total»);
3. User — хранит в себе id и фамилию пользователя (колонка «LastName»), имя пользователя (колонка «Name»), отчество пользователя (колонка «Patronymic»), логин (колонка «Login») и пароль (колонка «Password»);
4. IncomeOrExpenses —хранит в себе id и тип транзакции (колонка «Name»);
5. TypeIncome — хранит в себе id и тип дохода (колонка «Name»);
6. TypeExpenses —хранит в себе id и тип расхода (колонка «Name»).

1.5 Используемые технологии

При разработке приложения использовался  объектно-ориентированный язык программирования — C#. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет  и поддерживает:

1. статическую типизацию;
2. полиморфизм;
3. перегрузку операторов (в том числе операторов явного и неявного приведения типа);
4. делегаты;
5. атрибуты;
6. события;
7. переменные;
8. свойства;
9. обобщённые типы и методы;
10. итераторы, анонимные функции с поддержкой замыканий;
11. LINQ;
12. Исключения;
13. комментарии в формате XML.

С#‎, разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов С#‎, которая отражает BCL.

Также при разработке WPF приложения был использован инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL — «My Sql Workbench». Использовался SQL (англ. structured query language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. В общем случае SQL (без ряда современных расширений) считается языком программирования не полным по Тьюрингу, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

1. создание в базе данных новой таблицы;
2. добавление в таблицу новых записей;
3. изменение записей;
4. удаление записей;
5. выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
6. изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Также на площадке Visual Studio были использованы nuget-пакеты:

1. «Microsoft.EntityFrameworkCore.SqlServer» — в соответствии с рисунком 4;

https://cdn.discordapp.com/attachments/1107579711342784532/1108737018923581460/image.png

Рисунок 4

1. «Microsoft.EntityFrameworkCore.Tools» — в соответствии с рисунком 5.

https://cdn.discordapp.com/attachments/1107579711342784532/1108737065627164682/image.png

Рисунок 5

2 Описание разработанной системы

2.1 Описание интерфейса пользователя

В соответствии с приложением Б, на странице приложения в центре расположены кнопки «Зарегистрироваться» и «Войти»:

1. Кнопка «Зарегистрироваться» позволяет создать новый аккаунт для которого необходимо заполнить поля «Фамилия», «Имя», «Логин», «Пароль»;
2. Кнопка «Войти» позволяет войти в ранее созданный аккаунт, для этого нужно повторно ввести «Логин» и «Пароль» к которым ранее был привязан аккаунт в соответствующие поля.

В соответствии с приложением В, на главной странице приложения в левом верхнем углу доступны кнопки «Выйти» и «Статистика»:

1. Кнопка «Выйти» позволяет выйти из текущего аккаунта пользователя;
2. Кнопка «Статистика» открывает окно в котором пользователь может увидеть все свои транзакции и общую статистику.
3. Кнопка «Выбрать» запоминает выбор в выпадающем списке «Доходы/Расходы» и в зависимости от выбора создает один из выпадающих списков («Тип дохода» или «Тип расхода»);
4. Кнопка «Добавить» создает в статистике запись о новой транзакции.

Также в соответствии с приложением В, открывая приложение, пользователь попадает на главный экран, где расположены выпадающие списки:

1. Доходы/Расходы;
2. Тип дохода (в случае выбора «Доход» в выпадающем списке «Доходы/Расходы»);
3. Тип расхода (в случае выбора «Расход» в выпадающем списке «Доходы/Расходы»).

Также в соответствии с приложением В, открывая приложение, пользователь попадает на главный экран, где расположено поле для ввода:

1. Сумма.

2.2 Пользовательские сценарии

Сценарий работы с приложением:

Пользователю необходимо открыть WPF приложение и в соответствии с приложением Б пользователь попадет на окно авторизации, затем он должен будет нажать на кнопку «Зарегистрироваться», и в соответствии с приложением А пользователь окажется на странице регистрации

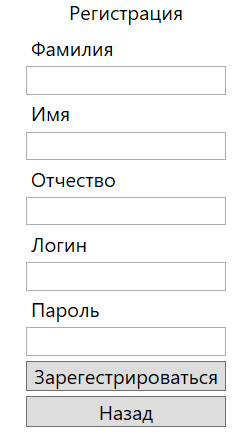


Рисунок 6

После этого в соответствии с приложением Б пользователь должен войти в ранее созданный аккаунт, введя придуманные логин и пароль в соответствующие поля.

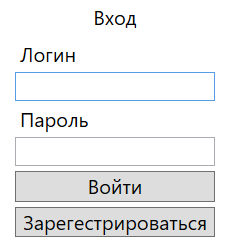


Рисунок 7

Далее пользователя в соответствии с приложением В перенесет на главную страницу в которой нужно будет выбрать «Доход» в выпадающем списке «Доходы/Расходы».

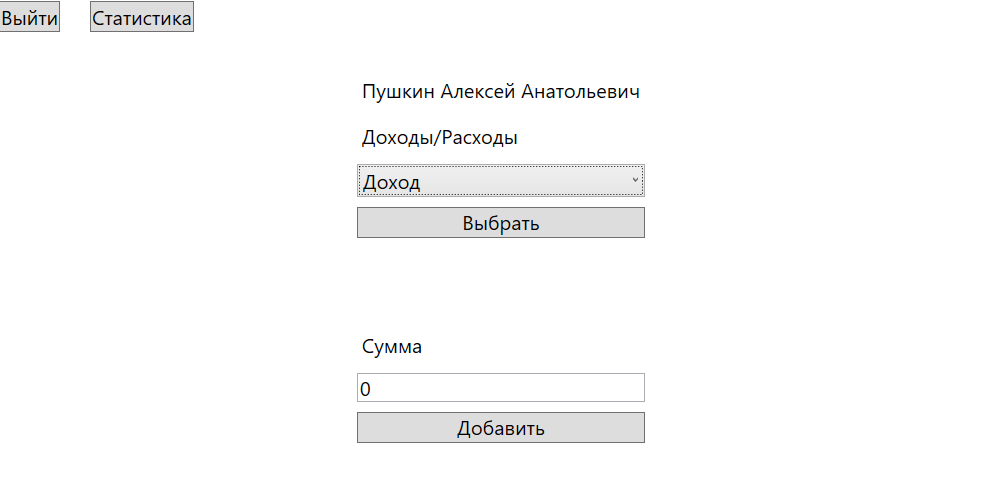


Рисунок 8

В ином случае приложение выдаст предупреждение и автоматически выберет «Доход».

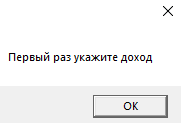


Рисунок 9

Затем нужно нажать кнопку «Выбрать» и выбрать в выпадающем списке тип дохода в соответствующем списке

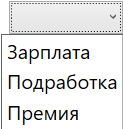


Рисунок 10

Далее ввести в поле ваш нынешний баланс «Сумма» и нажать кнопку «Добавить», чтобы создать вашу первую запись о поступлении средств на счет

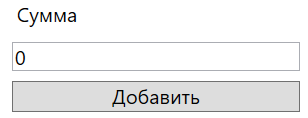


Рисунок 11

Далее можно будет выбрать «Расход» в выпадающем списке «Доходы/Расходы». Затем нужно нажать кнопку «Выбрать» и выбрать в выпадающем списке тип расхода в соответствующем списке

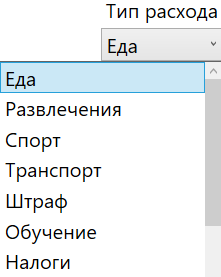


Рисунок 12

Далее ввести в поле ваш нынешний баланс «Сумма» и нажать кнопку «Добавить», чтобы создать новую запись о списании средств со счета, после чего приложение выдаст такое окно

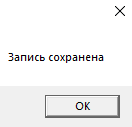


Рисунок 13

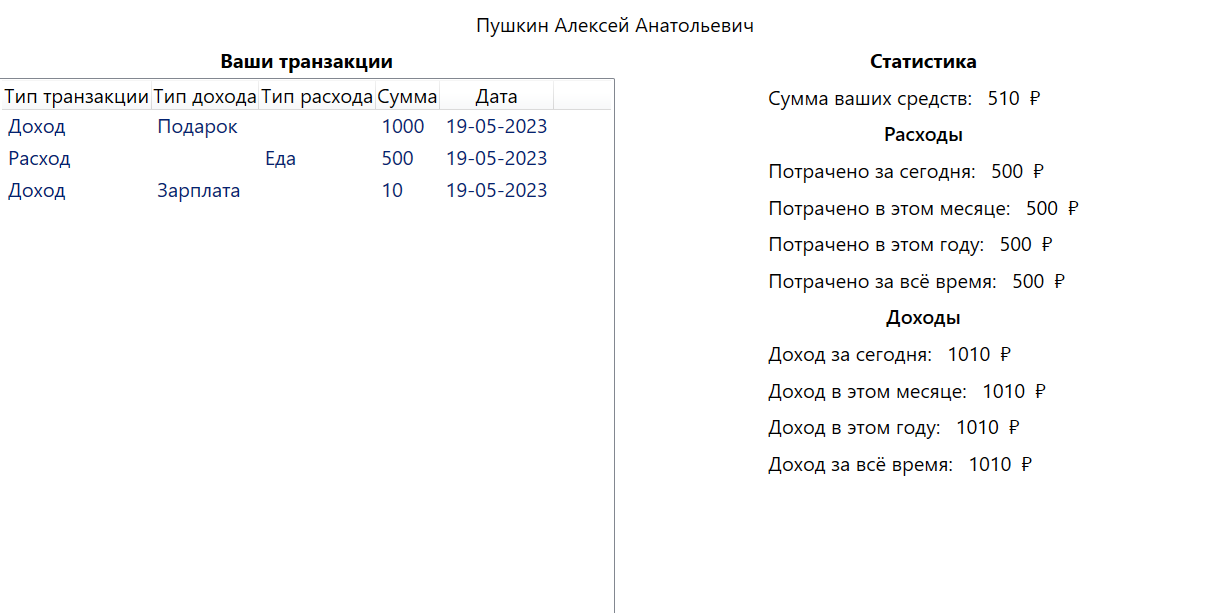
Также в соответствии с приложением Г можно в любой момент работы с главной страницей приложения нажать на кнопку «Статистика» в левом верхнем углу приложения, благодаря чему пользователя переместит на страницу со всей статистикой текущего аккаунта. Например: 

Рисунок 14

Заключение

При разработке WPF приложения удалось сделать форму добавления доходов и расходов и сохранение их в базе данных для последующего их использования. Удалось сделать просмотр добавленных транзакций и их даты. Также для удобства интерфейса удалось реализовать кнопку «Выйти», которая переносит пользователя на страницу авторизации, чтобы было удобнее переключаться между аккаунтами. При разработке WPF приложения не получилось сделать сортировку в окне статистики, а также удаление и редактирование данных в статистике. Исходя из цели курсового проекта, можно сказать, что главная цель достигнута, так как с помощью данного WPF приложения пользователь может сохранять и просматривать добавленные транзакции и общую статистику.

Список используемых источников

**Ресурсы сети интернет**

1. Общие сведения о Visual Studio / [Электронный ресурс]. <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019>, дата 14.05.2023.
2. C Sharp / [Электронный ресурс]. <https://ru.wikipedia.org/wiki/C_Sharp>, дата 15.05.2023.
3. Language Integrated Query / [Электронный ресурс]. <https://ru.wikipedia.org/wiki/Language_Integrated_Query>, дата 15.05.2023.
4. Common Language Runtime / [Электронный ресурс]. <https://ru.wikipedia.org/wiki/Common_Language_Runtime>, дата 15.05.2023
5. SQL / [Электронный ресурс]. <https://ru.wikipedia.org/wiki/SQL>, дата 16.05.2023.
6. Класс TimeLine / [Электронный ресурс]. <https://professorweb.ru/my/WPF/graphics_and_animation/level15/15_6.php>.
7. C# - Hide image when click outside of image in wpf / [Электронный ресурс]. <https://stackoverflow.com/questions/20238613/hide-image-when-click-outside-of-image-in-wpf>, дата 16.05.2023.
8. Диаграмма деятельности / [Электронный ресурс]. <https://ru.wikipedia.org/wiki/Диаграмма_деятельности>, дата 23.08.2023.

**Учебники и учебные пособия**

1. Троелсен. Эндрю. Джепикс. Филипп Язык программирования C# 7 и платфоромы .NET и .NET Core. 8-e изд : Пер с англ – СПБ : ООО «Диалектика», 2018 – 1328 с.

Приложение А  
Страница регистрации

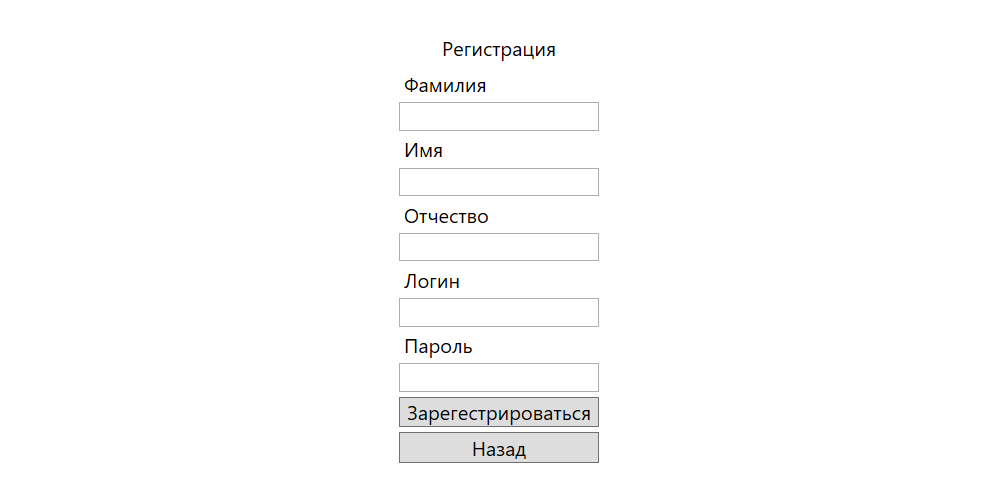


Рисунок 15

Приложение Б  
Страница авторизации

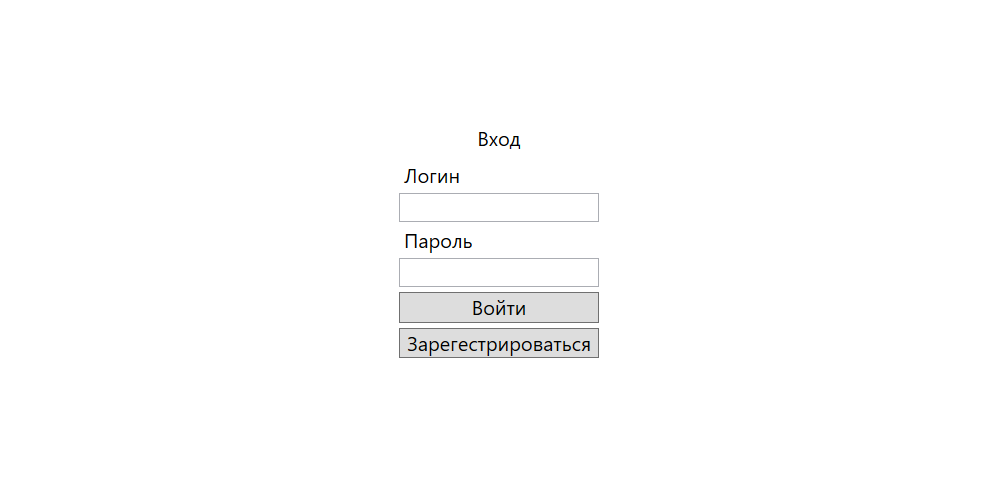


Рисунок 16

Приложение В

Главное окно WPF приложения

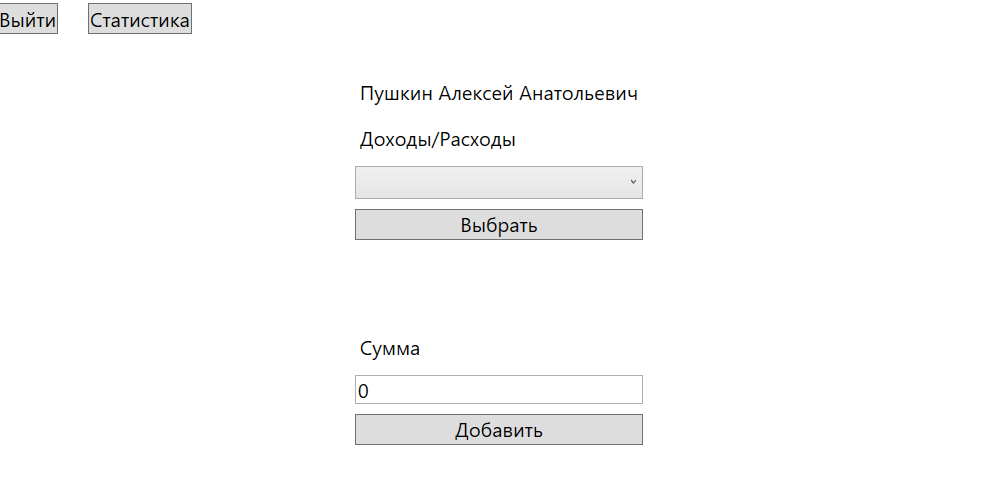


Рисунок 17

Приложение Г

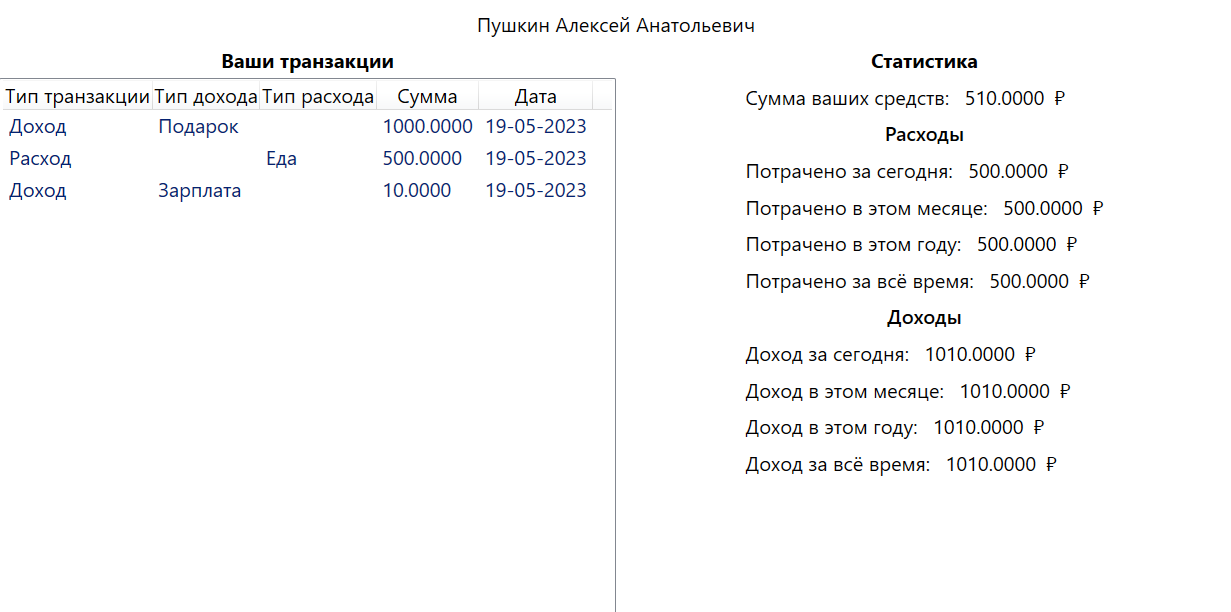
Окно статистики  


Рисунок 18

Лист нормоконтроля

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Нормоконтроль

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (и.о. фамилия)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.