**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

**Разработка информационной системы «Учет обмена валют»**

Выполнил студент гр. ИСП-22 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Красковский Артём Владимирович/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2025

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 3**](#_Toc195263484)

[**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4**](#_Toc195263485)

[**1.1 Платформа .NET 4**](#_Toc195263486)

[**1.2 Язык программирования С# 6**](#_Toc195263487)

[**1.3 Windows Presentation Foundation (WPF) 8**](#_Toc195263488)

[**1.4 СУБД SQL server 10**](#_Toc195263489)

[**1.5 Microsoft SQL Server Management Studio 12**](#_Toc195263490)

[**1.6 Entity Framework 13**](#_Toc195263491)

[**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 16**](#_Toc195263492)

[**2.1 Разработка диаграммы ERD 16**](#_Toc195263493)

[**2.2 Разработка базы данных 17**](#_Toc195263494)

[**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 26**](#_Toc195263495)

[**3.1 Разработка прототипа информационной системы 26**](#_Toc195263496)

[**3.2 Программирование информационной системы 29**](#_Toc195263497)

[**3.2.1 Разработка модуля «Авторизация» 29**](#_Toc195263498)

[**3.2.2 Разработка модуля «Курсы валют» 32**](#_Toc195263499)

[**3.2.3 Разработка модуля «Валюты» 34**](#_Toc195263500)

[**3.2.4 Разработка модуля «Операции» 37**](#_Toc195263501)

[**3.2.5 Разработка модуля «Отчеты» 45**](#_Toc195263502)

[**ЗАКЛЮЧЕНИЕ 49**](#_Toc195263503)

[**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 50**](#_Toc195263504)

# **ВВЕДЕНИЕ**

Актуальность выбранной темы заключается в том, что в современных условиях возрастает потребность в оперативном и точном учете валютных операций. Развитие финансовых технологий и рост объемов международных расчетов делают автоматизированные системы учета обмена валют необходимым инструментом для бизнеса. Такие системы позволяют оперативно отслеживать курсы валют, фиксировать сделки и минимизировать риски, связанные с ручным вводом данных.

Автоматизированная информационная система «Учет обмена валют» должна включать функции авторизации сотрудников, ввода и обработки данных о курсах валют, ведения журнала операций, формирования отчетов, а также контроля доступа и защиты информации. Это позволит упростить процесс учета операций, повысить точность данных и сократить время на формирование отчетности.

Данная система предназначена для кассиров по обмену валют и бухгалтеров, так как ее ключевыми преимуществами являются автоматизация учета операций, оперативное получение актуальных курсов валют и прозрачность отчетности. Эти функции помогут оптимизировать работу сотрудников, снизить количество ошибок и повысить эффективность финансовых операций, что в итоге усилит конкурентоспособность компании на рынке.

**Объект:** информационная система «Учет обмена валют»

**Предмет:** автоматизированная информационная система «Учет обмена валют»

**Цель работы:** разработать автоматизированную информационную систему «Учет обмена валют»

**Задачи:**

* выбрать инструментарий;
* спроектировать базу данных;
* разработать информационную систему.

# **ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

## **1.1 Платформа .NET**

Платформа .NET представляет собой комплексную среду разработки, созданную Microsoft для построения разнообразных приложений. Изначально задуманная как Windows-ориентированное решение, она трансформировалась в полноценную кроссплатформенную экосистему с открытым исходным кодом. Современная версия .NET поддерживает разработку веб-приложений, мобильных решений, облачных сервисов, игр и IoT-устройств, предлагая единый набор инструментов для различных сценариев использования.

Архитектура .NET основана на нескольких ключевых компонентах. Common Language Runtime (CLR) обеспечивает выполнение управляемого кода, предоставляя такие важные функции как управление памятью (включая автоматическую сборку мусора), безопасность типов и механизмы параллельного выполнения. Важной особенностью является система сборок, которая позволяет организовать код в логические единицы развертывания.

Библиотека базовых классов (BCL) содержит тысячи готовых классов и методов, охватывающих все основные задачи программирования - от работы с файлами и сетями до сложных математических вычислений. Особое внимание уделено асинхронному программированию, что критически важно для современных высоконагруженных приложений.

C# остается флагманским языком платформы, постоянно развиваясь и добавляя новые возможности. Последние версии языка включают такие функции как:

* Записи (records) для неизменяемых структур данных.
* Улучшенное сопоставление с образцом.
* Нативные поддержку nullable-ссылочных типов.
* Упрощенный синтаксис для работы с асинхронными операциями.

F# предлагает альтернативный функциональный подход, особенно полезный для сложных математических вычислений и обработки данных. Visual Basic.NET сохраняет поддержку для поддержки legacy-приложений, хотя его развитие существенно замедлилось.

Современная экосистема .NET предлагает разнообразные инструменты для разработки:

1. Visual Studio - полнофункциональная IDE с расширенными возможностями отладки и рефакторинга.

2. Visual Studio Code - легковесный редактор с богатой экосистемой расширений.

3. JetBrains Rider - кроссплатформенная альтернатива от JetBrains.

4. .NET CLI - мощный инструмент командной строки для всех этапов разработки.

Инструменты поддерживают современные практики разработки, включая:

* Интеграцию с системами контроля версий.
* Поддержку модульного тестирования.
* Инструменты профилирования и диагностики производительности.
* Возможности рефакторинга кода.

Платформа активно развивается, включая поддержку современных технологических трендов:

* ASP.NET Core для создания высокопроизводительных веб-приложений и API.
* Blazor позволяет использовать C# для клиентской веб-разработки.
* gRPC для эффективной меж процессной коммуникации.
* ML.NET для задач машинного обучения.
* .NET MAUI для кроссплатформенной разработки мобильных и десктопных приложений.

Для облачных решений .NET предлагает глубокую интеграцию с Azure, включая специализированные SDK для различных сервисов. Поддержка контейнеризации (Docker) и оркестрации (Kubernetes) делает платформу идеальным выбором для микро сервисной архитектуры.

Последние версии .NET демонстрируют впечатляющие показатели производительности благодаря:

* Улучшениям в JIT-компиляции.
* Оптимизациям работы сборщика мусора.
* Поддержке аппаратно-ускоренных операций.
* Эффективной работе с памятью.
* Улучшениям в библиотеках базовых классов.

Эти улучшения делают .NET конкурентоспособным решением для высоконагруженных систем, где критически важны показатели производительности и эффективности использования ресурсов.

## **1.2 Язык программирования С#**

C# (произносится как «си шарп») — это современный, объектно-ориентированный язык программирования, разработанный компанией Microsoft в 2000 году в рамках платформы .NET. Он сочетает в себе мощь и гибкость C++ с простотой и удобством Visual Basic. C# широко используется для разработки настольных, веб- и мобильных приложений, а также игр и облачных сервисов.

Язык C# был создан под руководством Андерса Хейлсберга, который также участвовал в разработке Turbo Pascal и Delphi. Первая версия языка вышла в 2000 году вместе с .NET Framework. С тех пор C# активно развивается, получая новые возможности с каждым обновлением. Важными вехами стали:

* C# 2.0 (2005) – добавлены generics, анонимные методы, nullable-типы.
* C# 3.0 (2007) – появились LINQ, лямбда-выражения, анонимные типы.
* C# 5.0 (2012) – поддержка асинхронного программирования через async/await.
* C# 9.0 (2020) – records, init-only свойства, улучшения pattern matching.
* C# 10.0 (2021) – глобальные using, file-scoped namespaces, улучшения структур.

C# продолжает развиваться, оставаясь одним из самых популярных языков в мире.

Основные особенности C#:

* Объектно-ориентированный подход – поддерживает инкапсуляцию, наследование, полиморфизм.
* Типизация – строгая статическая типизация с возможностью использования var для неявного определения типа.
* Автоматическое управление памятью – благодаря сборщику мусора (Garbage Collector).
* Кроссплатформенность – с появлением .NET Core (теперь .NET 5+) C# работает на Windows, Linux и macOS.
* Богатая стандартная библиотека – включает коллекции, файловый ввод-вывод, сетевые функции, многопоточность.
* Поддержка современных парадигм – функциональное программирование (LINQ, лямбды), асинхронность.

Области применения C#:

1. Разработка десктопных приложений.

C# широко используется для создания Windows-приложений с помощью фреймворков:

* Windows Forms – устаревшая, но простая технология для GUI.
* WPF (Windows Presentation Foundation) – современный подход с XAML и мощной графикой.
* MAUI (Multi-platform App UI) – кроссплатформенный фреймворк для Windows, macOS, iOS, Android.

1. Веб-разработка.

С помощью C# можно создавать серверные приложения и API:

* ASP.NET Core – высокопроизводительный фреймворк для веб-приложений и микро сервисов.
* Blazor – технология для создания интерактивных веб-интерфейсов на C# вместо JavaScript.

1. Разработка игр.

C# — основной язык для движка Unity, который используется в инди- и AAA-играх. Благодаря производительности и простоте на C# создают 2D- и 3D-игры для ПК, консолей и мобильных устройств.

1. Облачные технологии и микросервисы.

C# интегрируется с облачными платформами (Azure, AWS) и используется для:

* Разработки серверной логики.
* Создания RESTful API.
* Работы с базами данных через Entity Framework Core.

1. Мобильная разработка.

С помощью Xamarin (теперь часть .NET MAUI) можно писать нативные приложения для iOS и Android на C#.

## **1.3 Windows Presentation Foundation (WPF)**

WPF (Windows Presentation Foundation) - это современный фреймворк для создания графических интерфейсов в приложениях Windows. Разработанный Microsoft как часть .NET Framework, WPF пришел на смену Windows Forms, предлагая более мощные и гибкие возможности для построения пользовательских интерфейсов.

Основные особенности WPF:

* Использование XAML – интерфейс описывается декларативно в XML-подобном языке разметки.
* Разделение логики и дизайна – благодаря MVVM (Model-View-ViewModel) упрощается поддержка кода.
* Векторная графика и анимация – поддержка сложных визуальных эффектов без потери производительности.
* Привязка данных (Data Binding) – автоматическое обновление UI при изменении данных.
* Стили и шаблоны – возможность глобального изменения внешнего вида элементов.
* Аппаратное ускорение – рендеринг через DirectX для высокой производительности.

WPF построен на нескольких ключевых компонентах:

* PresentationFramework.dll – основные классы для работы с окнами, контролами и XAML.
* PresentationCore.dll – низкоуровневые элементы графики и ввода.
* Milcore.dll – интеграция с DirectX для рендеринга.
* WindowsBase.dll – базовые функции .NET, используемые WPF.

Графический движок WPF использует retained mode rendering, что позволяет эффективно управлять сложными сценами. WPF построен на основе композитной модели, где интерфейс состоит из визуальных элементов, организованных в древовидную структуру. Для размещения элементов используются различные типы панелей, каждая из которых реализует свою логику расположения дочерних компонентов.

Ядро WPF тесно интегрировано с DirectX, что обеспечивает аппаратное ускорение графики. Это позволяет создавать плавные анимации и сложные визуальные эффекты без существенной нагрузки на процессор.

При создании WPF-приложения разработчик обычно работает с двумя основными файлами: XAML-файлом, содержащим описание интерфейса, и файлом кода на C#, где реализуется логика работы. Типичное приложение состоит из окон (Window) или страниц (Page), содержащих различные элементы управления.

Пример простого окна может включать текстовое поле, кнопку и текстовый блок. При нажатии на кнопку текст из поля отображается в текстовом блоке. Такая простая реализация демонстрирует базовые принципы работы WPF, включая обработку событий и работу с элементами интерфейса.

WPF особенно хорошо подходит для создания бизнес-приложений со сложными интерфейсами, графических редакторов и других программ, где важны богатые возможности визуализации. Его главными преимуществами являются гибкость, мощная система привязки данных и поддержка современного дизайна.

Однако WPF требует больше ресурсов, чем Windows Forms, и имеет более высокий порог входа для начинающих разработчиков. Тем не менее, для создания профессиональных Windows-приложений WPF остается одним из лучших выборов среди технологий .NET.

## **1.4 СУБД SQL server**

SQL Server представляет собой мощную систему управления базами данных, разработанную Microsoft для предприятий любого масштаба. Эта СУБД сочетает в себе высокую производительность с надежностью и безопасностью, что делает ее популярным выбором для бизнес-приложений.

В основе SQL Server лежит реляционная модель данных с поддержкой транзакций, обеспечивающая целостность информации. Система использует специализированный диалект SQL — Transact-SQL (T-SQL), который расширяет стандартные возможности языка дополнительными функциями и операторами.

Современные версии SQL Server предлагают комплексный подход к работе с данными. Помимо хранения информации, система включает инструменты для бизнес-аналитики, интеграции данных и создания отчетов. Особенностью последних выпусков стала глубокая интеграция с облачными технологиями и поддержка работы с большими данными.

SQL Server построен по клиент-серверной модели, где ядро системы отвечает за обработку запросов и управление данными. Важной составляющей является оптимизатор запросов, который анализирует и улучшает выполнение SQL-инструкций. Система автоматически управляет распределением ресурсов, обеспечивая эффективную работу даже при больших нагрузках.

Для администрирования и разработки Microsoft предоставляет SQL Server Management Studio — многофункциональную среду, позволяющую управлять серверами, разрабатывать запросы и настраивать производительность. Альтернативой для кроссплатформенной работы является Azure Data Studio.

Разработчики ценят SQL Server за стабильность и предсказуемость работы. Например, при создании веб-приложения на ASP.NET Core часто используют эту СУБД как основное хранилище данных. Типичный сценарий включает создание базы данных, таблиц и хранимых процедур, которые затем вызываются из кода приложения.

В корпоративной среде SQL Server часто выступает в роли центрального хранилища данных, интегрируясь с другими бизнес-системами. Возможности репликации и резервного копирования обеспечивают отказоустойчивость критически важных приложений.

Современные тенденции развития SQL Server включают усиленную поддержку работы в гибридных средах, где данные могут находиться как в локальных дата-центрах, так и в облаке. Особое внимание уделяется безопасности данных и соответствию различным стандартам защиты информации. В новых версиях появляются функции для работы с графовыми данными и машинного обучения непосредственно в среде базы данных.

## **1.5 Microsoft SQL Server Management Studio**

SQL Server Management Studio (SSMS) давно зарекомендовал себя как основной инструмент для взаимодействия с SQL Server. Эта интегрированная среда объединяет в себе все необходимые функции для администрирования, разработки и анализа производительности баз данных.

SSMS предоставляет разработчикам и администраторам удобный графический интерфейс для выполнения повседневных задач. С его помощью можно не только писать и выполнять SQL-запросы, но и полностью управлять инфраструктурой сервера. Особенно ценными являются встроенные средства для анализа планов выполнения запросов, позволяющие находить узкие места в работе базы данных.

Интеллектуальный редактор кода с подсветкой синтаксиса и авто дополнением значительно ускоряет процесс разработки. При этом среда поддерживает работу с различными версиями SQL Server, включая облачные решения Azure SQL Database.

Типичная сессия работы начинается с подключения к серверу через интуитивно понятный диалог аутентификации. После установки соединения пользователь получает доступ к древовидной структуре всех объектов базы данных через обозреватель объектов. Это позволяет легко ориентироваться в сложных схемах и быстро находить нужные таблицы, представления или хранимые процедуры.

Для выполнения запросов разработчики используют многофункциональное окно редактора, где можно не только писать код, но и анализировать результаты его выполнения. Особенно полезной оказывается возможность просматривать статистику выполнения и визуализировать план запроса, что помогает в оптимизации производительности.

Помимо основных функций работы с базами данных, SSMS включает множество специализированных инструментов. Среди них стоит отметить мощную систему мониторинга активности сервера, средства для настройки безопасности и управления пользователями, а также удобные мастера для выполнения операций обслуживания.

Для администраторов особенно ценными являются возможности централизованного управления несколькими серверами, инструменты для настройки репликации и функции аварийного восстановления. Все это делает SSMS незаменимым инструментом в арсенале любого специалиста по работе с SQL Server.

## **1.6 Entity Framework**

Entity Framework (EF) — это объектно-ориентированная технология доступа к данным, разработанная Microsoft. Она представляет собой ORM (Object-Relational Mapping) фреймворк, который позволяет разработчикам работать с реляционными базами данных, используя объекты и классы вместо прямого написания SQL-запросов. Entity Framework существенно упрощает взаимодействие с данными, автоматизируя многие рутинные задачи, такие как маппинг сущностей, управление соединениями и выполнение CRUD-операций.

Одним из ключевых преимуществ Entity Framework является его способность абстрагировать базу данных, позволяя разработчикам сосредоточиться на бизнес-логике приложения, а не на деталях хранения данных. EF поддерживает различные подходы к работе с данными, включая Database First, Model First и Code First. Эти подходы дают гибкость в проектировании структуры данных и позволяют выбирать наиболее подходящий вариант в зависимости от требований проекта.

Entity Framework интегрируется с различными СУБД, включая Microsoft SQL Server, MySQL, PostgreSQL и SQLite, благодаря поддержке провайдеров данных. Это делает его универсальным инструментом для разработки приложений, работающих с разными системами управления базами данных. Кроме того, EF поддерживает LINQ (Language Integrated Query), что позволяет писать запросы к данным прямо в коде на C# или VB.NET с помощью удобного синтаксиса.

Entity Framework состоит из нескольких ключевых компонентов, которые обеспечивают его функциональность. Центральным элементом является DbContext — класс, который представляет сеанс работы с базой данных. Он отвечает за отслеживание изменений в сущностях, управление соединениями и выполнение запросов. DbContext также предоставляет доступ к DbSet — коллекциям сущностей, которые соответствуют таблицам в базе данных.

Другим важным компонентом является модель данных, которая может быть создана автоматически на основе существующей базы данных (Database First), сгенерирована из визуальной модели (Model First) или определена в коде с помощью классов (Code First). Вне зависимости от подхода, Entity Framework автоматически создает маппинг между объектами и таблицами базы данных, избавляя разработчика от необходимости писать SQL-скрипты вручную.

Entity Framework также включает механизм миграций, который позволяет управлять изменениями в структуре базы данных. Миграции дают возможность последовательно применять изменения схемы данных, сохраняя согласованность между кодом и базой данных на разных этапах разработки. Это особенно полезно в командной работе и при развертывании приложений в разных средах.

Использование Entity Framework имеет ряд преимуществ. Во-первых, он значительно сокращает объем кода, необходимого для работы с базой данных, благодаря автоматизации многих процессов. Во-вторых, поддержка LINQ позволяет писать более читаемые и безопасные запросы, поскольку ошибки могут быть выявлены на этапе компиляции. В-третьих, EF обеспечивает кроссплатформенность благодаря .NET Core и совместимости с различными СУБД.

Однако у Entity Framework есть и некоторые недостатки. Производительность сложных запросов может быть ниже, чем у нативного SQL, из-за дополнительного уровня абстракции. Кроме того, в некоторых случаях автоматически сгенерированные SQL-запросы оказываются неоптимальными, что требует ручной настройки или использования хранимых процедур. Также изучение всех возможностей EF может занять время, особенно для начинающих разработчиков.

# **ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

## **2.1 Разработка диаграммы ERD**

ER-диаграмма (схема «сущность-связь») — это визуальное представление структуры базы данных. Основными компонентами ER-диаграммы являются сущности, атрибуты и связи. Сущности представляют собой объекты или понятия, которые имеют значение в контексте системы. Атрибуты описывают свойства сущностей. Связи показывают, как сущности взаимодействуют друг с другом. ER-диаграммы широко используются в проектировании баз данных для организации информации и оптимизации процессов.

Связи между сущностями могут быть различными: один-к-одному, один-ко-многим и многие-ко-многим. Эти связи определяют, как данные одной сущности соотносятся с данными другой.

Нотация "Crows Foot" (или "воронья лапка") — это графический способ представления связей между сущностями на ER-диаграммах. Она получила свое название из-за характерного символа, напоминающего след вороньей лапки, который используется для обозначения множественности связей. В этой нотации каждая связь между сущностями изображается линией, на концах которой могут быть различные символы, указывающие на тип и характер связи.

ER-диаграммы могут быть созданы на разных уровнях детализации:

* Концептуальный уровень - отражает общее понимание предметной области.
* Логический уровень - более детальный уровень, на котором определяются все необходимые сущности и связи.
* Физический уровень - самый детальный уровень, соответствующий конкретной базе данных.

ER-диаграммы (диаграммы "сущность-связь") играют важную роль в моделировании и визуализации структуры данных в системах управления базами данных. Они помогают наглядно представить сущности и их взаимосвязи, что упрощает понимание системы. Кроме того, ER-диаграммы служат основой для проектирования реляционных баз данных, позволяя определить необходимые таблицы и поля.

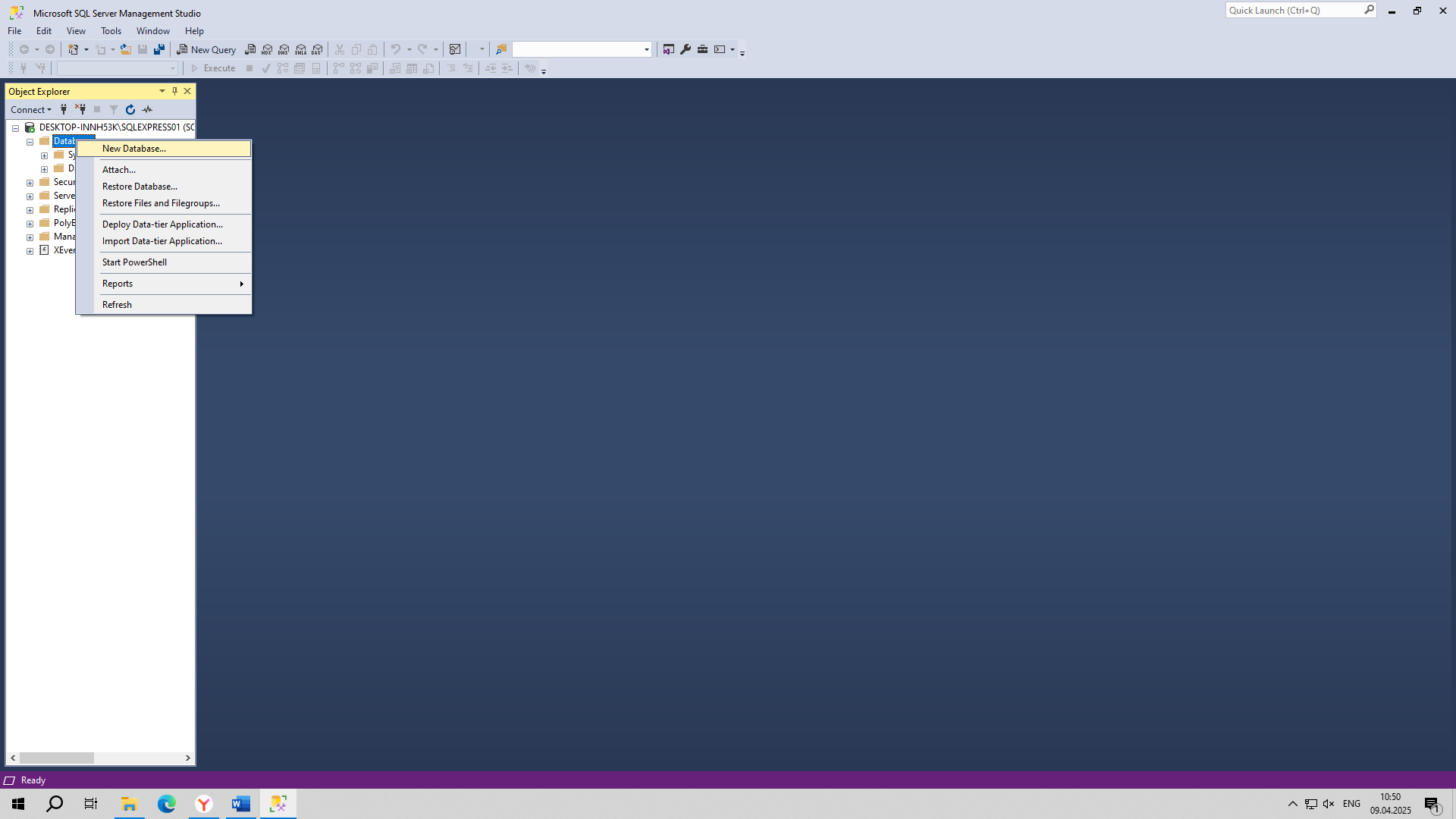


*Рис 2.1.1 ER-диаграмма «Учет обмена валют»*

## **2.2 Разработка базы данных**

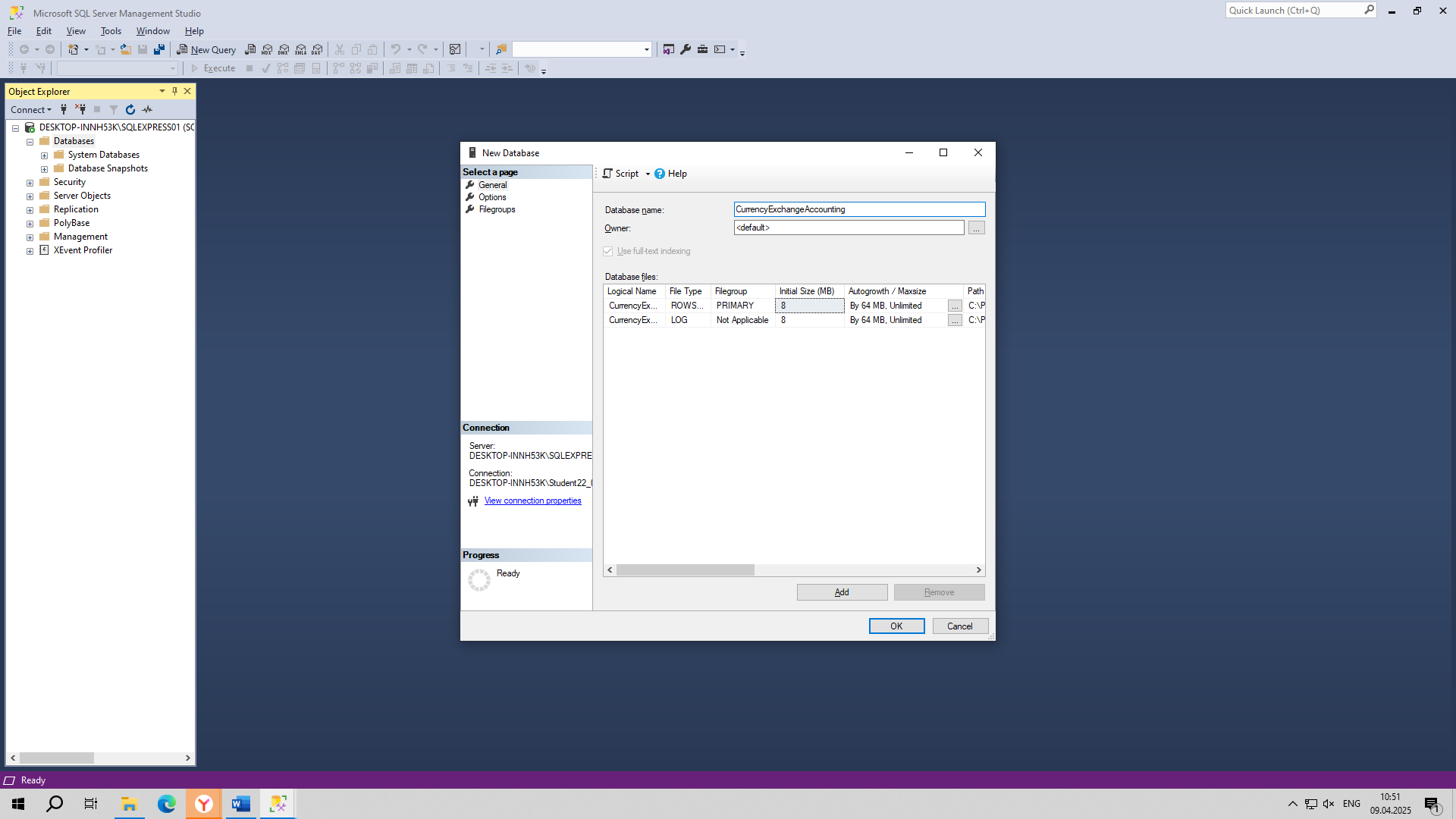
SQL Server Management Studio (SSMS) давно зарекомендовал себя как основной инструмент для взаимодействия с SQL Server. Эта интегрированная среда объединяет в себе все необходимые функции для администрирования, разработки и анализа производительности баз данных.

SSMS предоставляет разработчикам и администраторам удобный графический интерфейс для выполнения повседневных задач. С его помощью можно не только писать и выполнять SQL-запросы, но и полностью управлять инфраструктурой сервера. Особенно ценными являются встроенные средства для анализа планов выполнения запросов, позволяющие находить узкие места в работе базы данных.



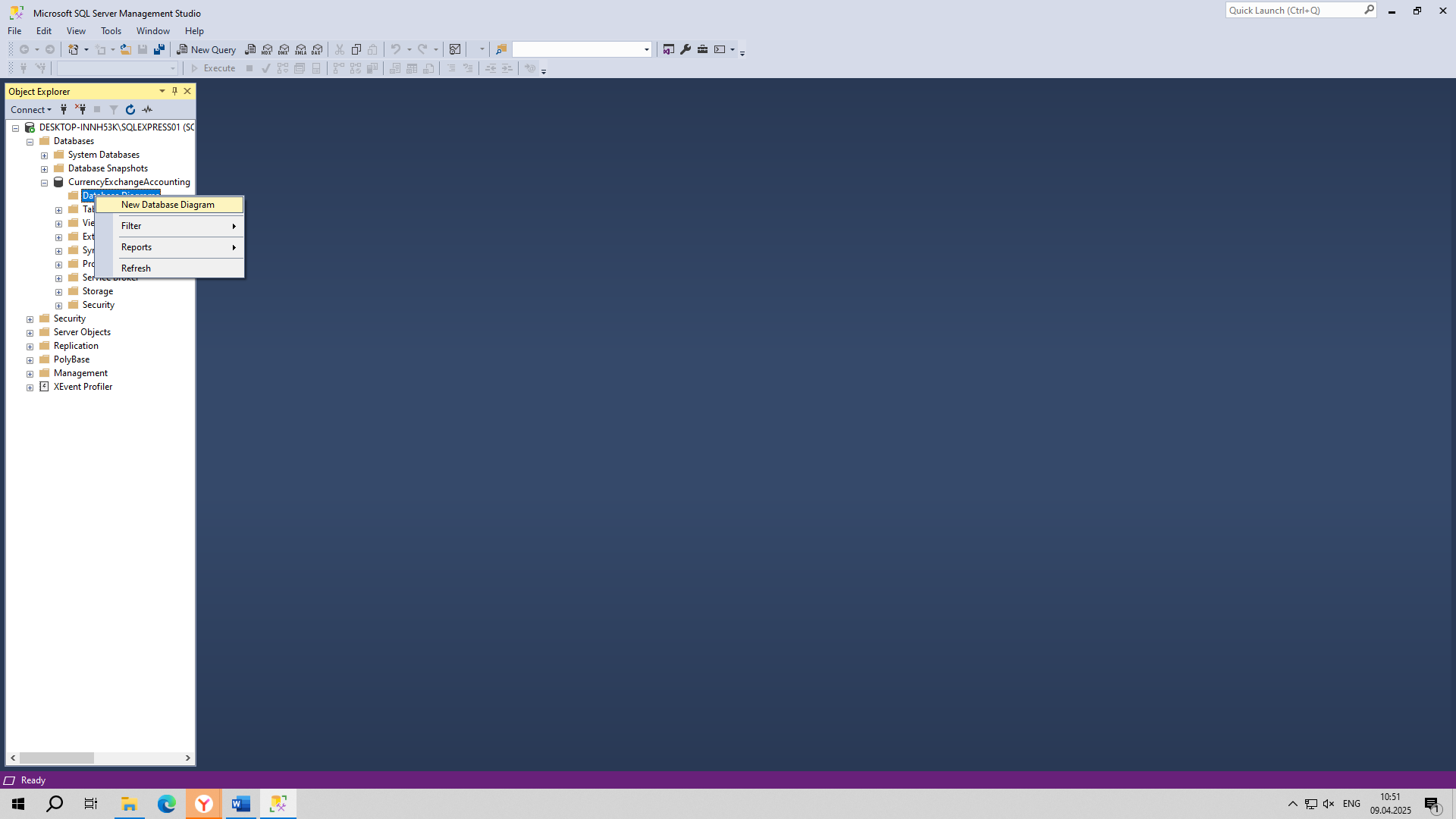
*Рис. 2.2.1 Создание базы данных*

На изображении показан интерфейс Microsoft SQL Server Management Studio (SSMS) с открытым меню управления базами данных. Интерфейс позволяет создавать базы данных, таблицы, а также управлять резервным копированием и восстановлением.



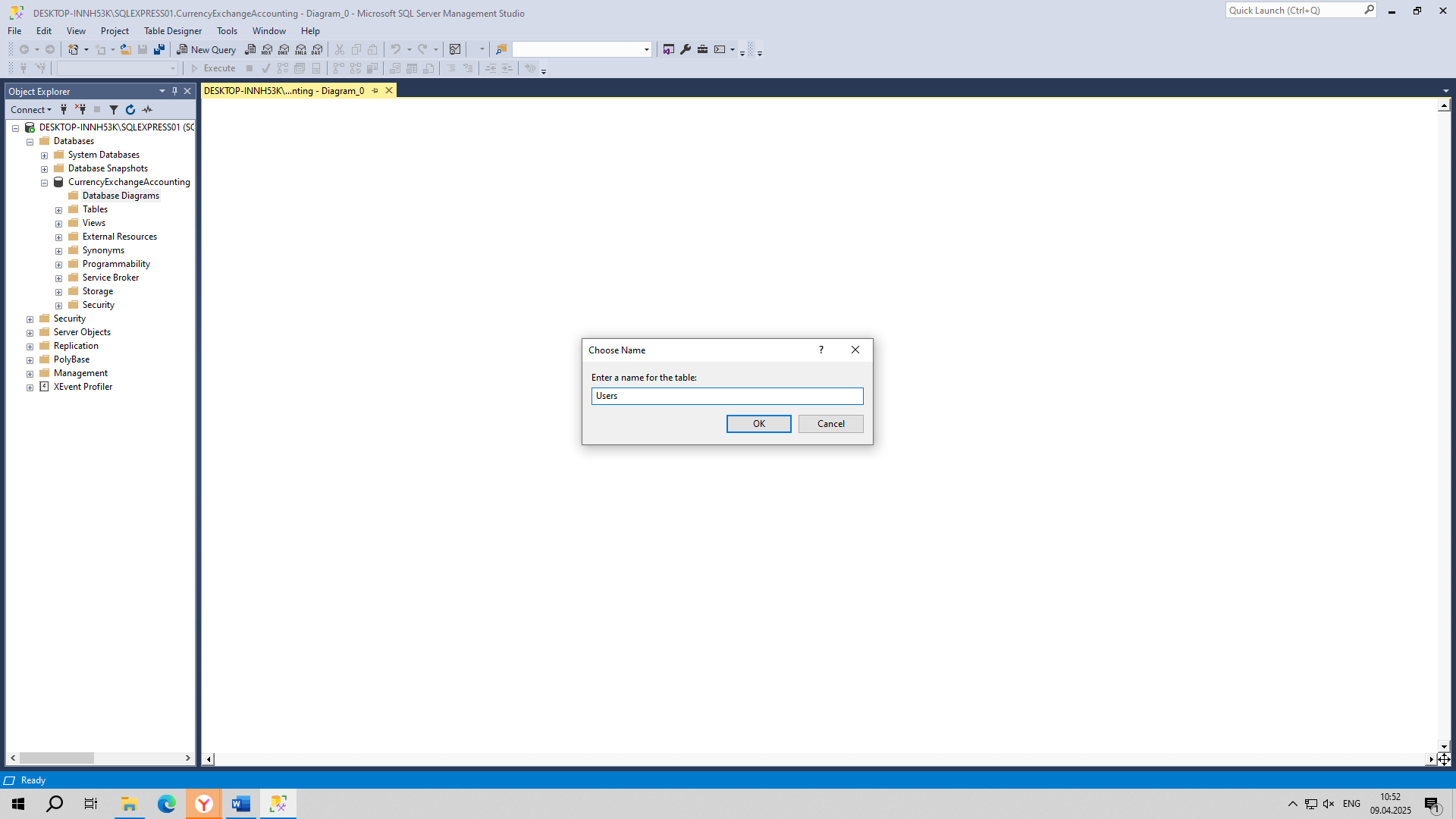
*Рис. 2.2.2 Создание базы данных «CurrencyExchangeAccounting»*

На изображении представлено окно создания новой базы данных в Microsoft SQL Server Management Studio. Основные параметры базы данных включают имя «CurrencyExchnageAccounting» и владельца по умолчанию.



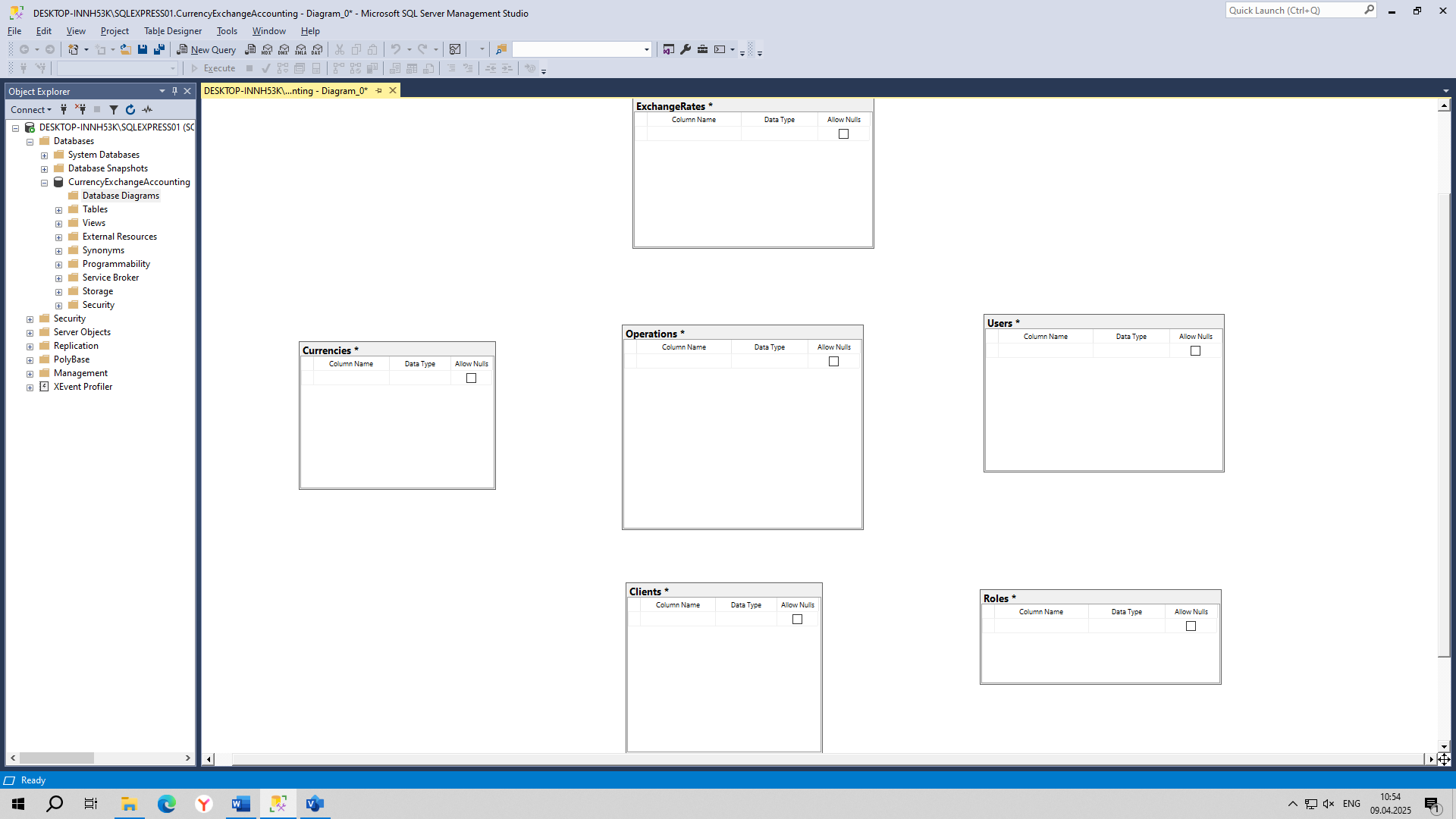
*Рис 2.2.3 Создание диаграммы базы данных*

На изображении показан интерфейс создания диаграммы базы данных в Microsoft SQL Server Management Studio (SSMS). В левой части экрана видно меню Object Explorer с развернутым разделом выбранной базы данных «CurrencyExchangeAccounting». Этот инструмент позволяет визуально проектировать структуру базы данных, отображая таблицы, их поля, связи и ключи в графическом виде.



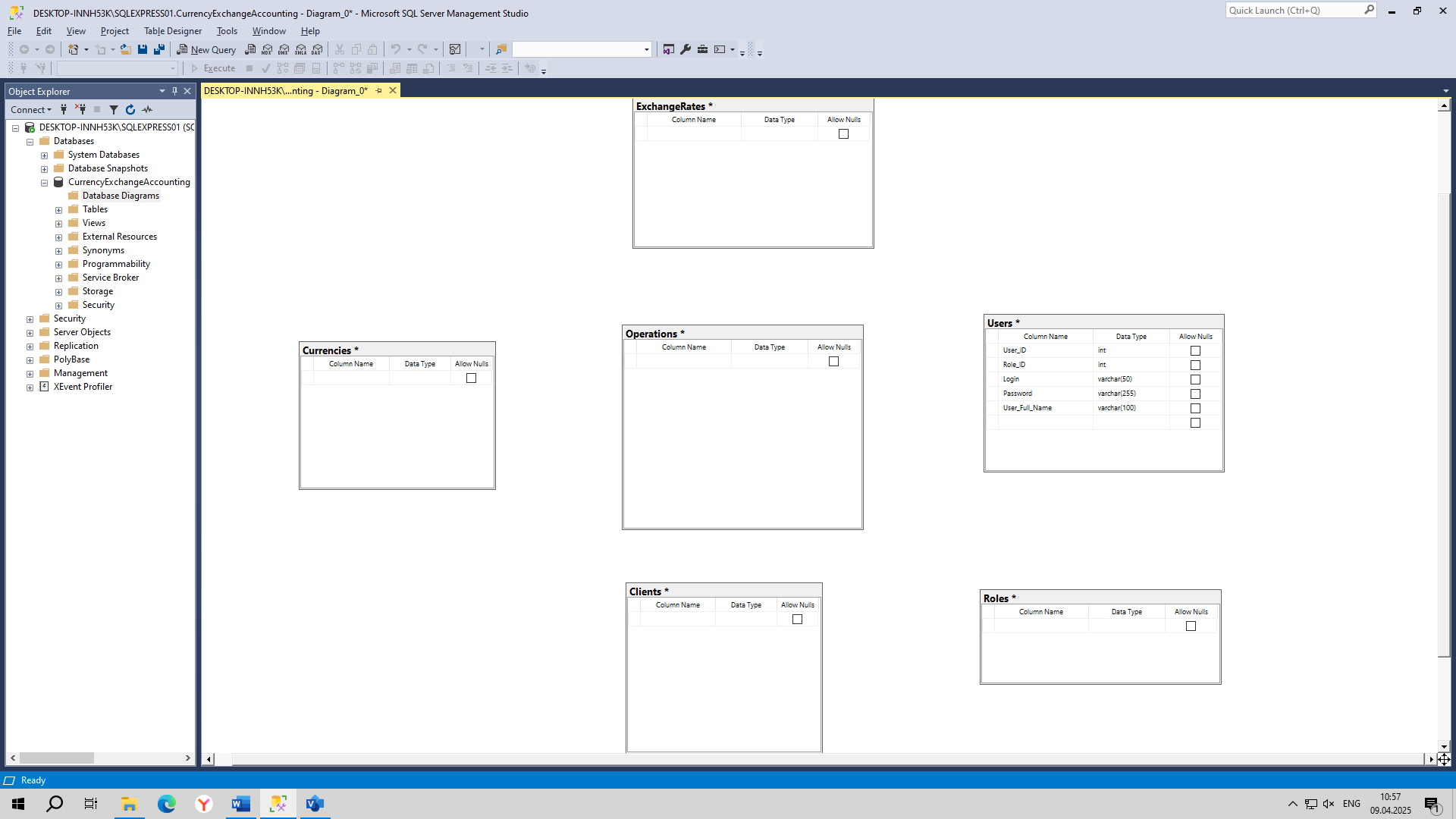
*Рис. 2.2.4 Создание таблицы «Users»*

На изображении представлено диалоговое окно Microsoft SQL Server Management Studio для создания новой таблицы в режиме проектирования диаграммы базы данных.



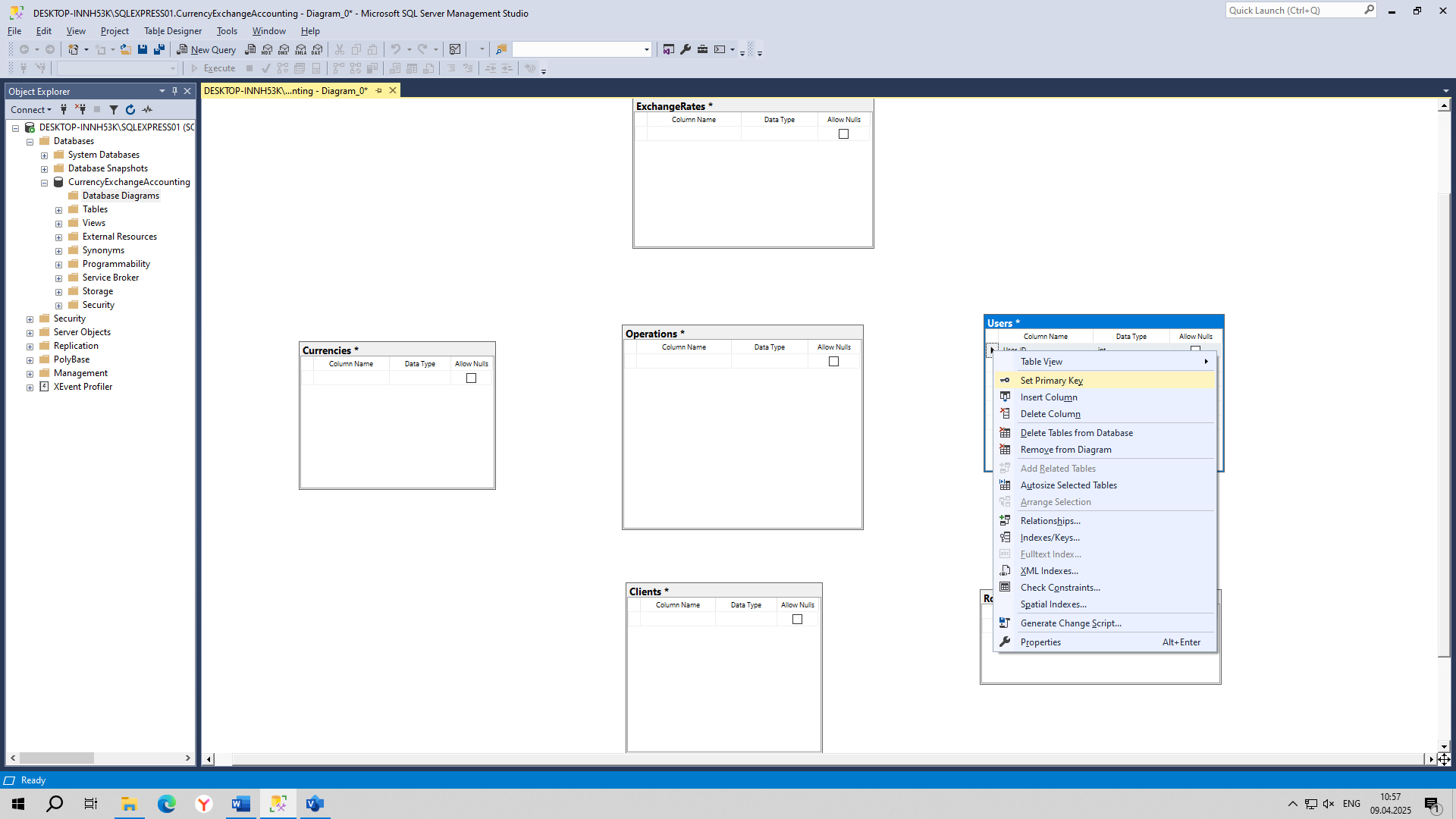
*Рис 2.2.5 Были созданы таблицы базы данных*

На изображении показан интерфейс проектирования базы данных в Microsoft SQL Server Management Studio, где представлена диаграмма с таблицами: ExchangeRates, Currencies, Operations, Users, Clients и Roles.



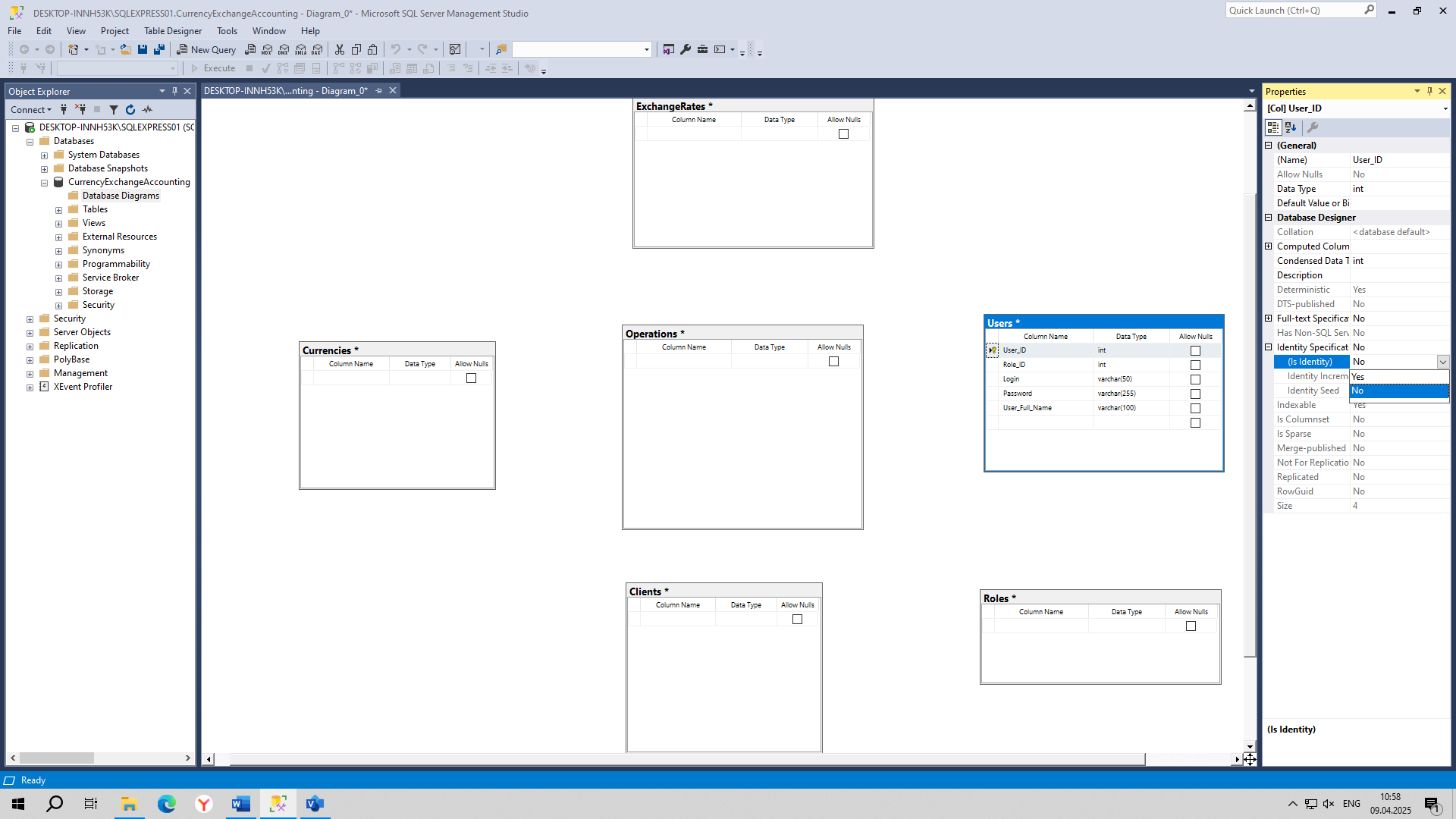
*Рис 2.2.6 Была заполнена таблица «Users»*

На изображении представлена таблица базы данных, заполненная в соответствии со структурой, определенной в ER-диаграмме.



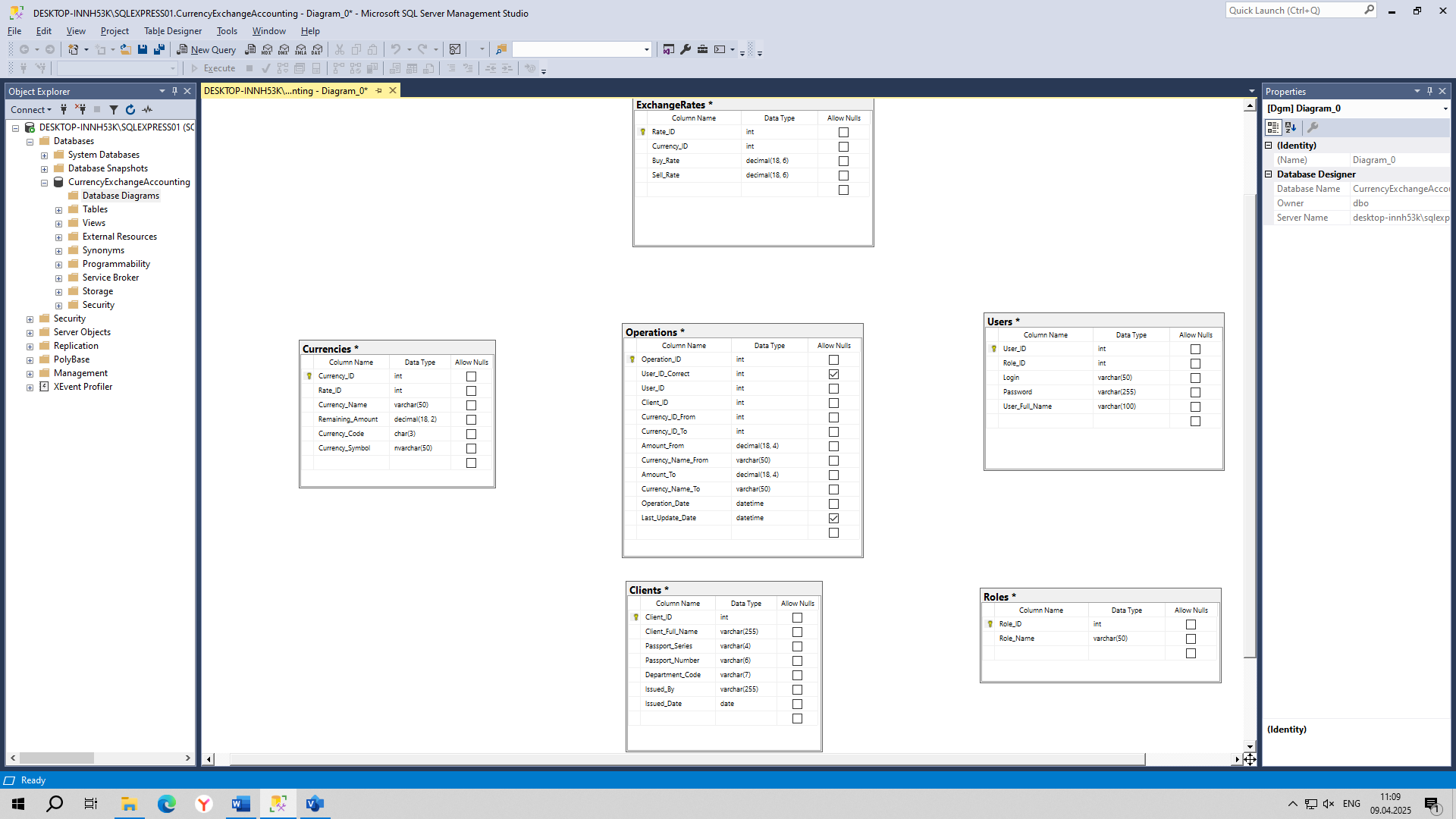
*Рис 2.2.7 Назначение первичного ключа*

На рисунке изображён интерфейс для управления таблицей «Users» в диаграмме базе данных. Пункт «Set Primary Key» позволяет определить первичный ключ для таблицы.



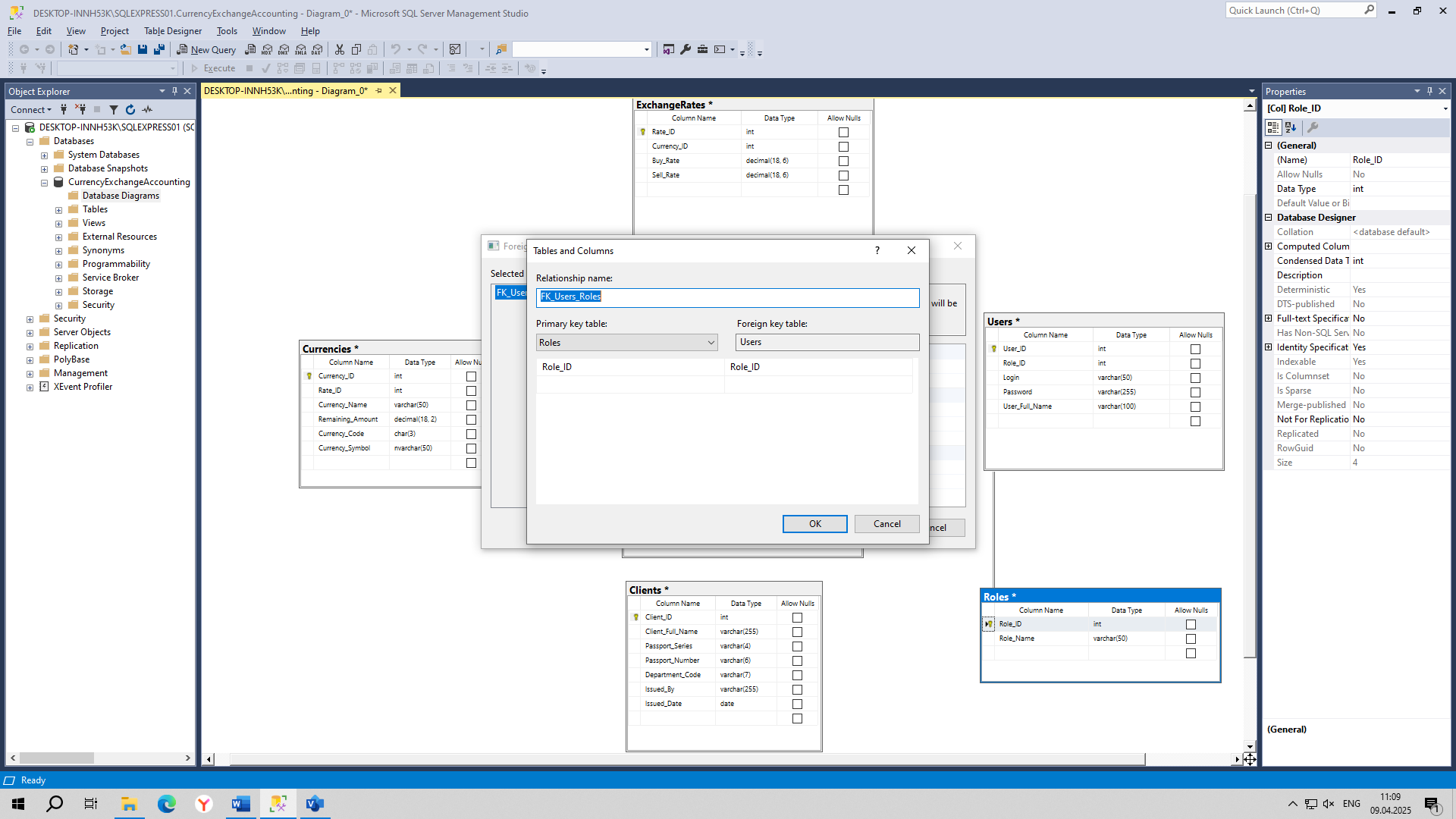
*Рис 2.2.8 Назначение автозаполнения*

На рисунке изображён интерфейс для управлением свойствами таблицы «Users» в диаграмме базе данных. Пункт «Identity Specification» позволяет назначить автозаполнение для колонны «User\_ID».



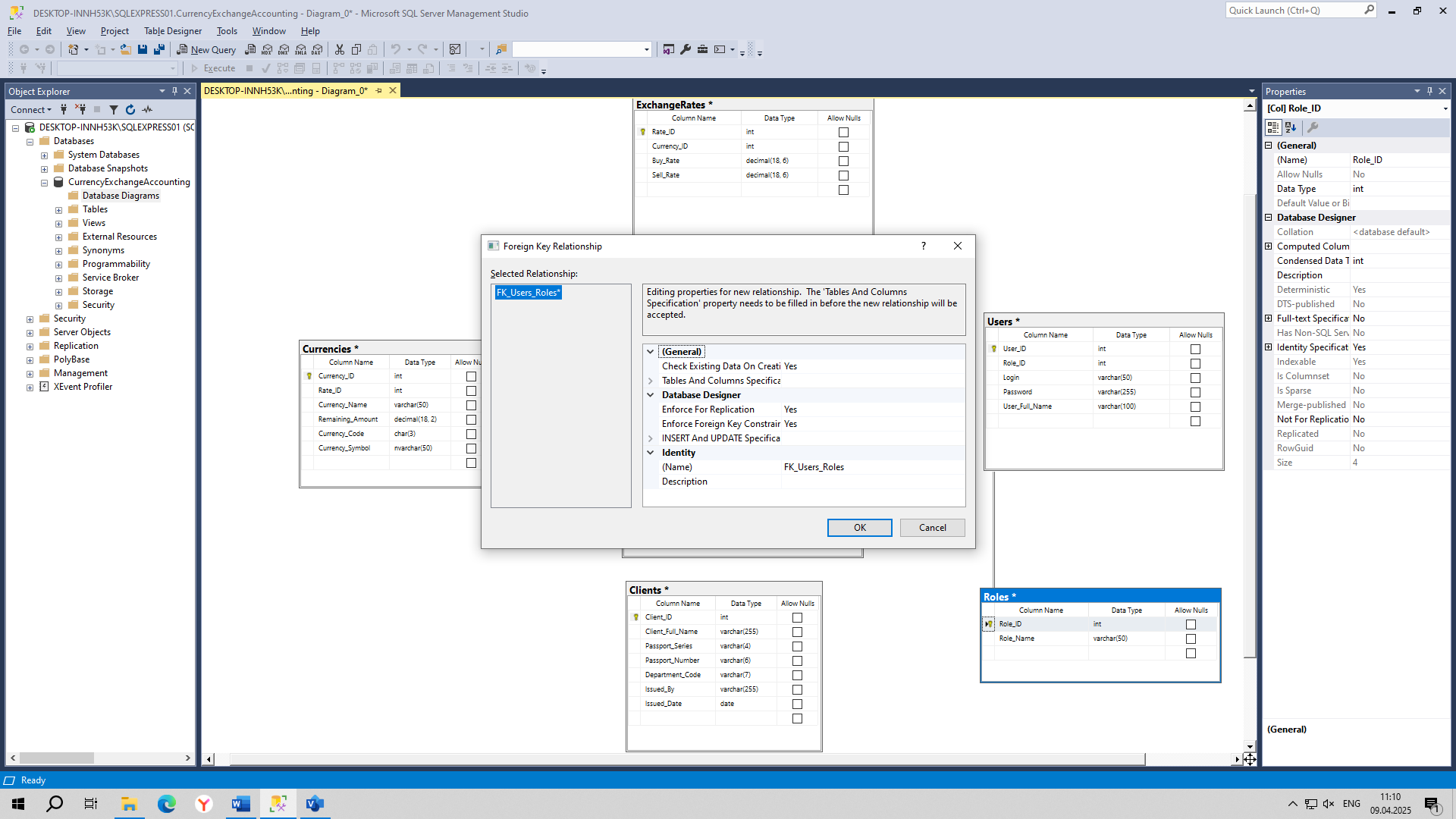
*Рис 2.2.9 Были заполнены все таблицы*

На изображении представлены таблицы базы данных, заполненные в соответствии со структурой, определенной в ER-диаграмме.



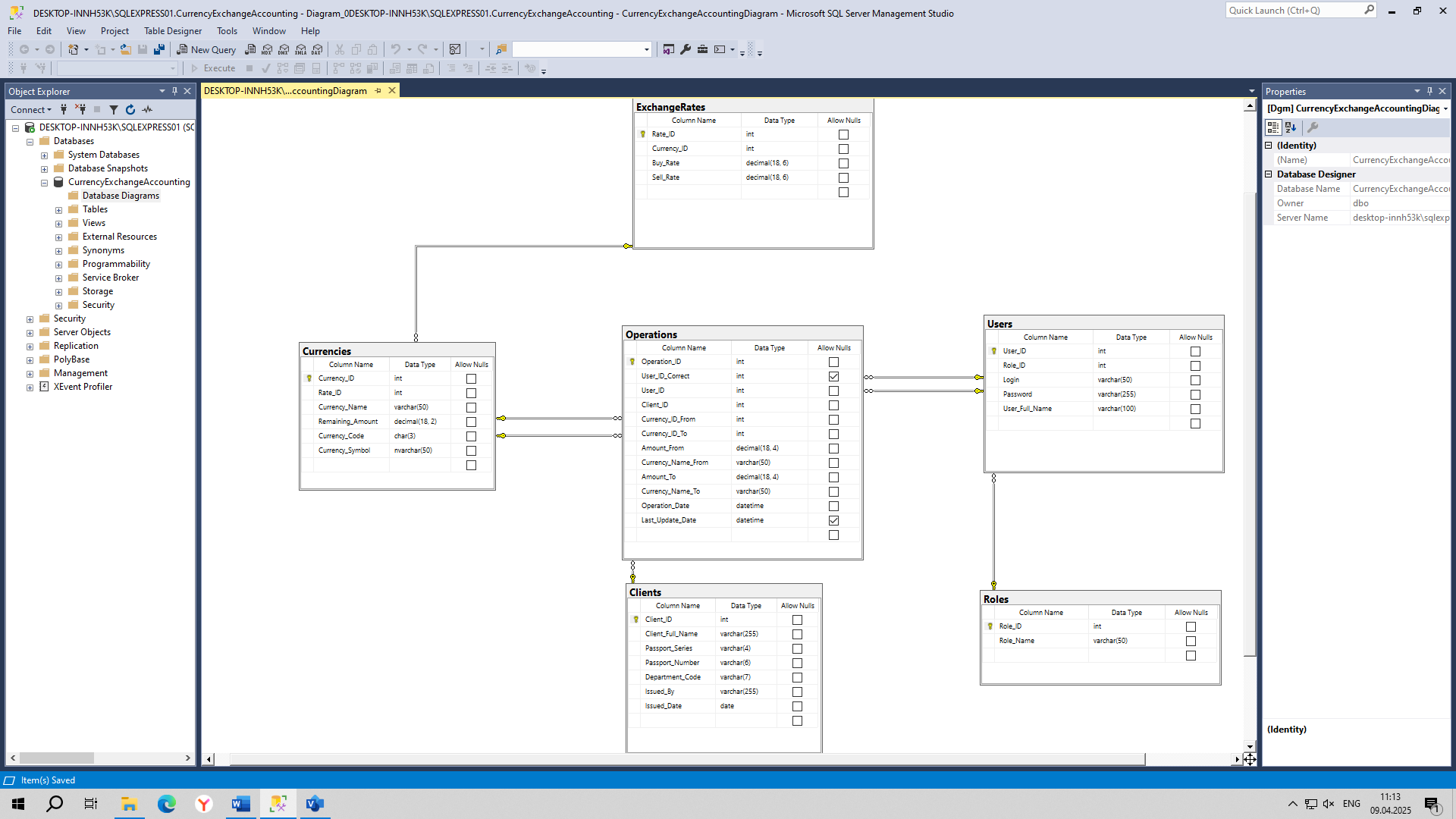
*Рис. 2.2.10 Создание связи между таблицами «Roles» и «Users»*

На изображении отображен интерфейс создания связи между таблицами базы данных.



*Рис. 2.2.11 Создание связи между таблицами «Roles» и «Users»*

На изображении отображен интерфейс создания связи между таблицами базы данных.



*Рис 2.2.12 Готовая диаграмма базы данных*

В данной работе используется 6 таблиц базы данных. Сущность users (Пользователи) имеет атрибуты: User\_ID (Уникальный ключ), Role\_ID (Внешний ключ), Login (Логин), Password (Пароль), User\_Full\_Name (ФИО пользователя). Сущность Roles (Роли) имеет атрибуты: Role\_ID (Уникальный ключ), Role\_Name (Название роли). Сущность ExchangeRates (Курсы обмена) имеет атрибуты: Rate\_ID (Уникальный ключ), Currency\_ID (Внешний ключ), Buy\_Rate (Курс покупки), Sell\_Rate (Курс продажи). Сущность Currencies (Валюты) имеет атрибуты: Currency\_ID (Уникальный ключ), Rate\_ID (Внешний ключ), Currency\_Name (Название валюты), Remaining\_Amount (Остаток валюты), Currency\_Code (Код валюты), Currency\_Symbol (Символ валюты). Сущность Clients (Клиенты) имеет атрибуты: Client\_ID (Уникальный ключ), Client\_Full\_Name (ФИО клиента), Passport\_Series (Серия пасспорта), Passport\_Number (Номер пасспорта), Department\_Code (Код подразделения), Issued\_By (Кем выдан), Issued\_Date (Дата выдачи). Сущность Operations (Операции) имеет атрибуты: Operations\_ID (Уникальный ключ), User\_ID\_Correct (Внешний ключ), User\_Id (Внешний ключ), Client\_ID (Внешний ключ), Currency\_ID\_From (Внешний ключ), Currency\_ID\_To (Внешний ключ), Amount\_From (Сумма продажи), Currency\_Name\_From (Валюта продажи), Amount\_To (Сумма покупки), Currency\_Name\_To (Валюта покупки), Operation\_Date (Дата операции), Last\_Update\_Date (Последнее изменение).

# **ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

## **3.1 Разработка прототипа информационной системы**



*Рис 3.1.1 Окно «Авторизация»*

На изображении отображен интерфейс окна «Авторизация» содержащее форму для ввода учетных данных с полями "Логин" и "Пароль". В нижней части интерфейса расположены две кнопки управления: "Отменить" для сброса введенных данных и "Вход" для подтверждения авторизации.



*Рис 3.1.2 Окно «Курсы валют»*

На изображении показан интерфейс окна «Курсы валют». В верхней части экрана расположены элементы управления – строка поиска и кнопку фильтрации данных. Центральную область занимает таблица с информацией о валютах.



*Рис 3.1.3 Окно «Валюты»*

На изображении показан интерфейс окна «Валюты». В верхней части экрана расположены поле поиска и кнопка фильтрации данных. Основное пространство окна занимает таблица с актуальными курсами валют.



*Рис 3.1.4 Окно «Операции»*

На изображении показан интерфейс окна «Операции». В верхней части экрана расположены поле для поиска, кнопка фильтрации данных и кнопка добавления новой операции. Центральную часть окна занимает таблица со списком всех операций.

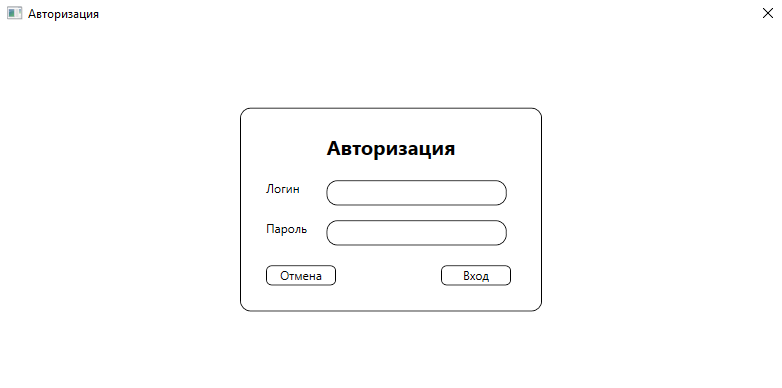


*Рис. 3.1.5 Окно «Отчеты»*

На изображении показан интерфейс окна «Отчеты». В верхней части экрана отображается поля выбора периода, кнопки управления отчетов. Основную часть окна занимает таблица с данными отчетов.

## **3.2 Программирование информационной системы**

### **3.2.1 Разработка модуля «Авторизация»**



*Рис. 3.2.1.1 Окно «Авторизация»*

Методы генерации капчи, проверки логина, пароля и капчи окна «Авторизация»:

// Обработчик нажатия клавиш в поле ввода кода

private void CodeTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter || e.Key == Key.Return) // Проверка на нажатие Enter

{

// Проверка совпадения кода и его срока действия

if (CodeTextBox.Text == \_generatedCode && DateTime.Now < \_codeExpirationTime)

{

\_codeTimer.Stop(); // Останавливаем таймер

string roleName = GetUserRoleName(\_currentUser.Role\_ID); // Получаем роль пользователя

MessageBox.Show($"Добро пожаловать, {\_currentUser.User\_Full\_Name}!\nВаша роль: {roleName}",

"Успешный вход", MessageBoxButton.OK, MessageBoxImage.Information);

// Открываем главное окно в зависимости от роли пользователя

Window mainWindow;

if (\_currentUser.Roles.Role\_Name == "Старший кассир")

{

mainWindow = new GeneralWindowSenior(\_currentUser); // Окно для старшего кассира

}

else

{

mainWindow = new GeneralWindowOperator(\_currentUser); // Окно для оператора

}

mainWindow.Show(); // Отображаем главное окно

this.Close(); // Закрываем текущее окно

}

else

{

MessageBox.Show("Неверный код подтверждения или время истекло", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

// Проверка логина в базе данных

private Users CheckUserLogin(string login)

{

using (var context = new CurrencyExchangeAccountingEntities()) // Создание контекста базы данных

{

return context.Users

.Include(u => u.Roles) // Подгрузка связанных данных о ролях

.FirstOrDefault(u => u.Login == login); // Поиск пользователя по логину

}

}

// Проверка пароля

private bool CheckPassword(Users user, string password)

{

return user.Password == password; // Проверка совпадения паролей

}

// Генерация случайного кода подтверждения

private string GenerateConfirmationCode()

{

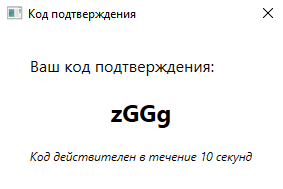
const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&\*";

var random = new Random();

return new string(Enumerable.Repeat(chars, 4) // Генерация строки из 4 символов

.Select(s => s[random.Next(s.Length)]).ToArray());

}



*Рис. 3.2.1.2 Окно «Код подтверждения»*

Метод окна «Код подтверждения»:

public CodeConfirmationWindow(string code)

{

InitializeComponent();

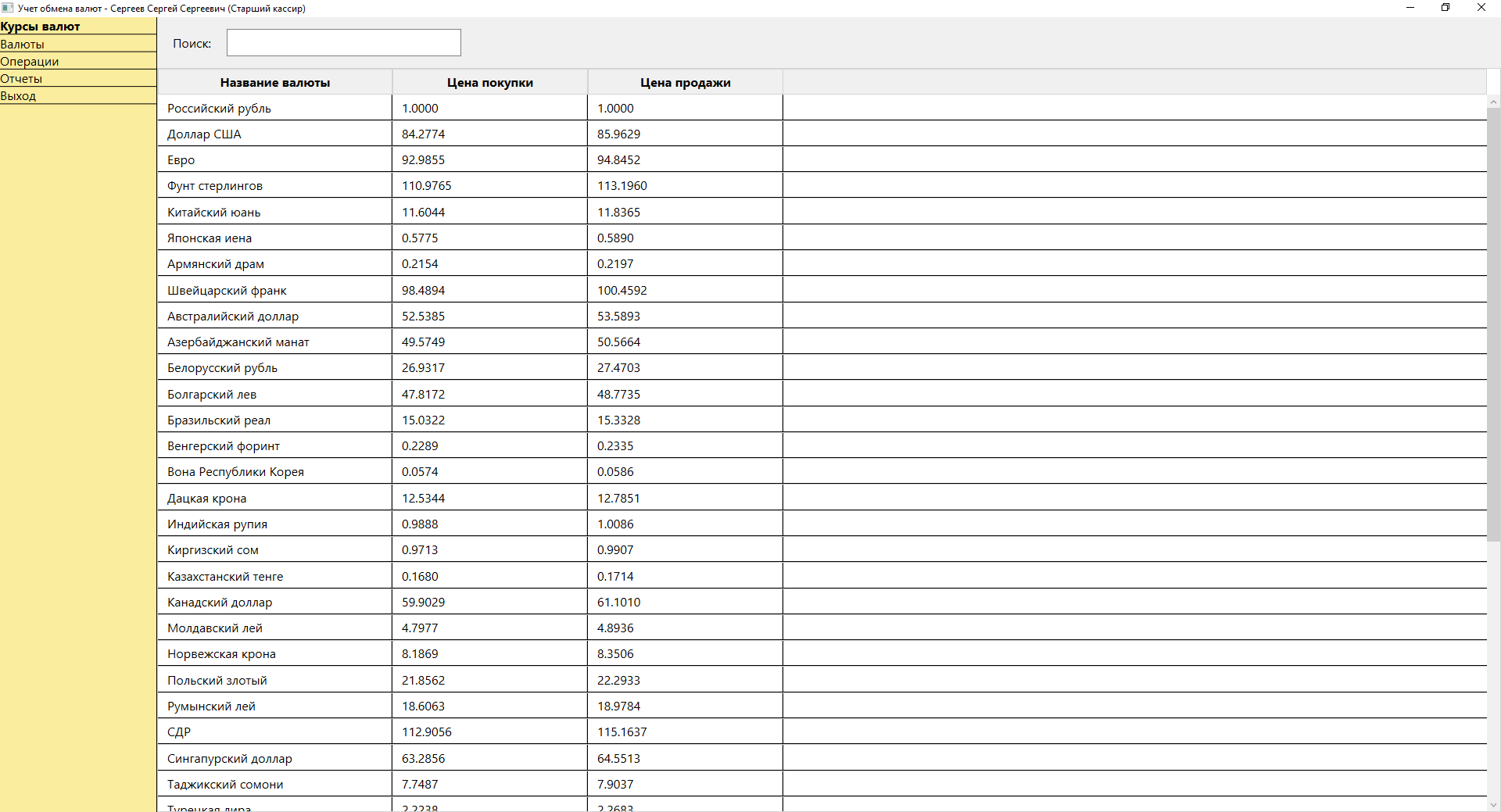
// Установка полученного кода в текстовый блок

CodeTextBlock.Text = code;

}

В данном модуле реализовано окно авторизации с несколькими уровнями проверки. При запуске отображается форма для ввода логина и пароля. После ввода логина система проверяет его существование в базе данных. Если логин существует, разблокируется поле для ввода пароля. После успешной проверки пароля генерируется код подтверждения, который пользователь должен ввести в отдельном окне. Если код введен правильно в течение 10 секунд, пользователь получает доступ к главному окну системы, где отображается его роль (оператор или старший кассир). При необходимости можно повторить отправку кода.

### **3.2.2 Разработка модуля «Курсы валют»**



*Рис 3.2.2.1 Окно «Курсы валют»*

Методы Загрузки данных из базы данных окна «Курсы валют»:

// Метод загрузки данных о валютах из базы данных.

private void LoadCurrencyData()

{

using (var context = new CurrencyExchangeAccountingEntities())

{

// Получение всех валют и их курсов из базы данных.

\_allCurrencies = (from c in context.Currencies

join r in context.ExchangeRates on c.Rate\_ID equals r.Rate\_ID

select new CurrencyData

{

Currency\_Name = c.Currency\_Name,

Buy\_Rate = r.Buy\_Rate,

Sell\_Rate = r.Sell\_Rate

}).ToList();

// Создание копии всех валют для фильтрации.

\_filteredCurrencies = new List<CurrencyData>(\_allCurrencies);

// Привязка отфильтрованных данных к таблице.

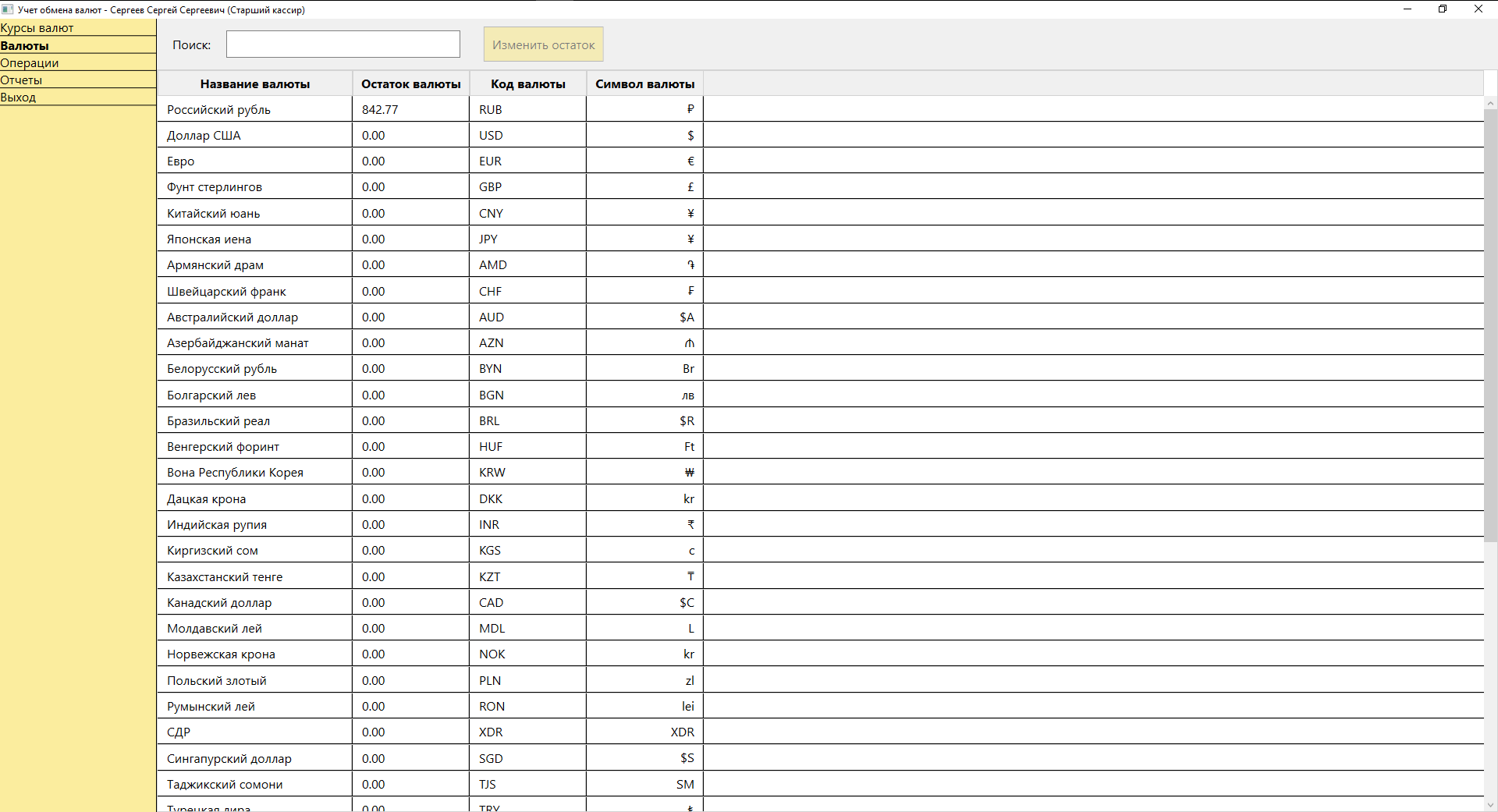
CurrenciesDataGrid.ItemsSource = \_filteredCurrencies;

}

}

В модуле реализовано окно для отображения курсов валют. При загрузке страницы данные о валютах (название, курс покупки, курс продажи) подгружаются из базы данных и отображаются в таблице. Пользователь может фильтровать валюты по названию через специальное поле поиска. Таблица поддерживает сортировку по любому столбцу (название, курс покупки, курс продажи) как по возрастанию, так и по убыванию. Данные в таблице отображаются с применением стилей, которые обеспечивают читаемость и удобство просмотра информации.

### **3.2.3 Разработка модуля «Валюты»**



*Рис 3.2.3.1 Окно «Валюты»*

Методы загрузки данных из базы данных и переход на страницу изменения остатков валюты окна «Валюты»:

// Метод для перехода на страницу редактирования баланса валюты.

private void NavigateToEditBalancePage()

{

// Если валюта не выбрана, показываем предупреждение.

if (\_selectedCurrency == null)

{

MessageBox.Show("Выберите валюту для изменения остатка", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Если навигационный сервис доступен, переходим на страницу редактирования.

if (NavigationService != null)

{

NavigationService.Navigate(new EditCurrencyBalancePageSenior(\_currentUser, \_selectedCurrency));

}

}

// Метод для загрузки данных о валютах из базы данных.

public void LoadCurrencyData()

{

// Создание контекста базы данных.

using (var context = new CurrencyExchangeAccountingEntities())

{

// Получение всех валют из базы данных.

\_allCurrencies = context.Currencies.ToList();

// Вывод информации о каждой валюте в консоль для отладки.

foreach (var currency in \_allCurrencies)

{

Debug.WriteLine($"Currency: {currency.Currency\_Code}, Symbol: {currency.Currency\_Symbol}");

}

// Инициализация отфильтрованного списка всеми валютами.

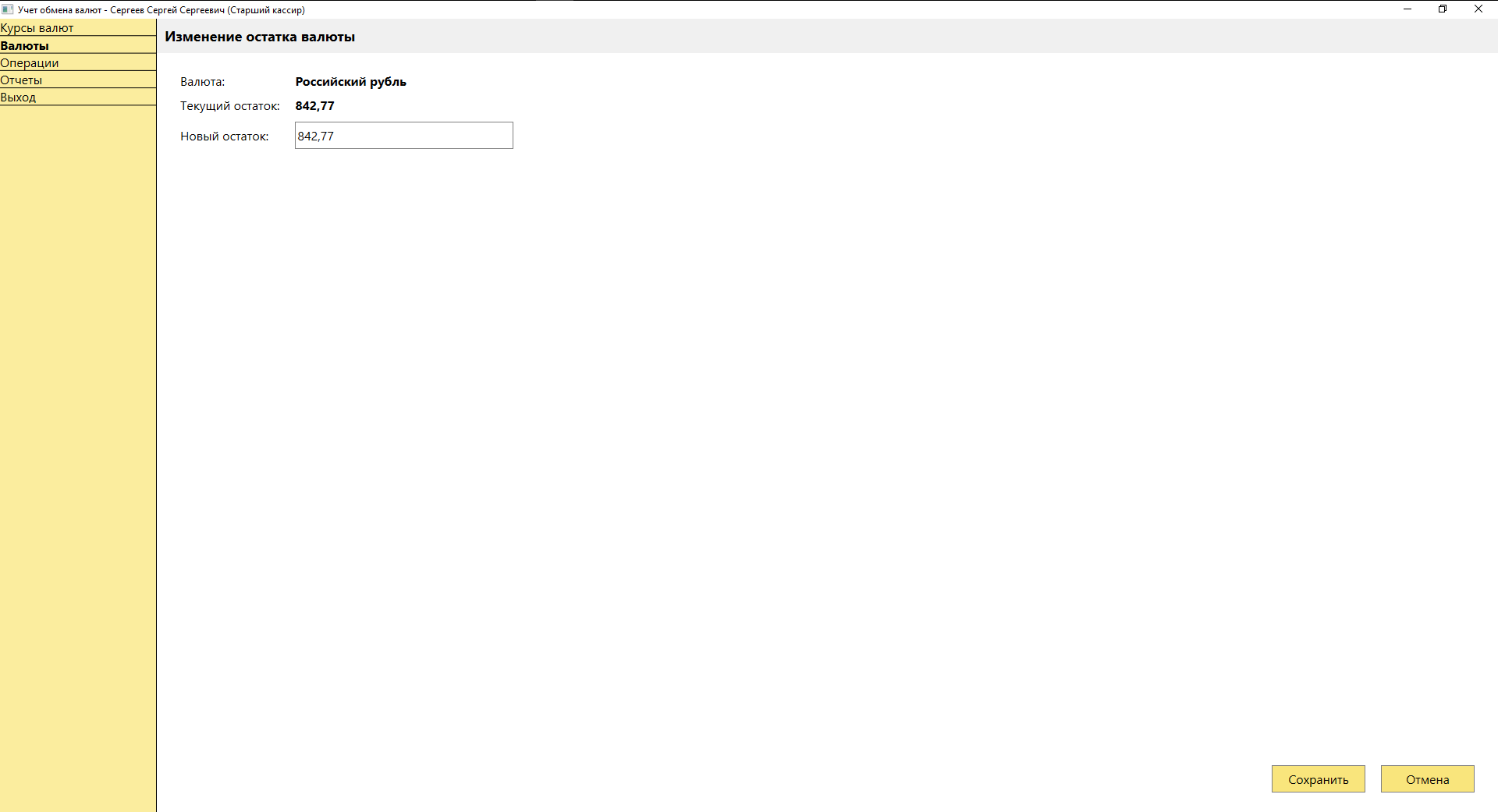
\_filteredCurrencies = new List<Currencies>(\_allCurrencies);

// Привязка отфильтрованного списка к таблице.

CurrenciesDataGrid.ItemsSource = \_filteredCurrencies;

}

}



*Рис 3.2.3.2 Окно «Изменение остатка валюты»*

Метод сохранения остатка валюты окна «Изменение остатка валюты»:

// Обработчик события нажатия кнопки "Сохранить"

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

// Проверка корректности ввода нового значения остатка

if (!decimal.TryParse(NewBalanceTextBox.Text, out decimal newBalance) || newBalance < 0)

{

MessageBox.Show("Введите корректное положительное число", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

// Поиск валюты в базе данных по её идентификатору

var currency = \_context.Currencies.Find(\_currencyId);

// Если валюта найдена, обновляем её остаток

if (currency != null)

{

currency.Remaining\_Amount = newBalance;

// Сохраняем изменения в базе данных

\_context.SaveChanges();

// Уведомление приложения о том, что данные валют были обновлены

App.RaiseCurrenciesUpdated();

// Отображение сообщения об успешном сохранении

MessageBox.Show("Остаток валюты успешно обновлен", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

// Безопасный возврат назад

SafeNavigateBack();

}

}

catch (Exception ex)

{

// Обработка исключений при сохранении данных

MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

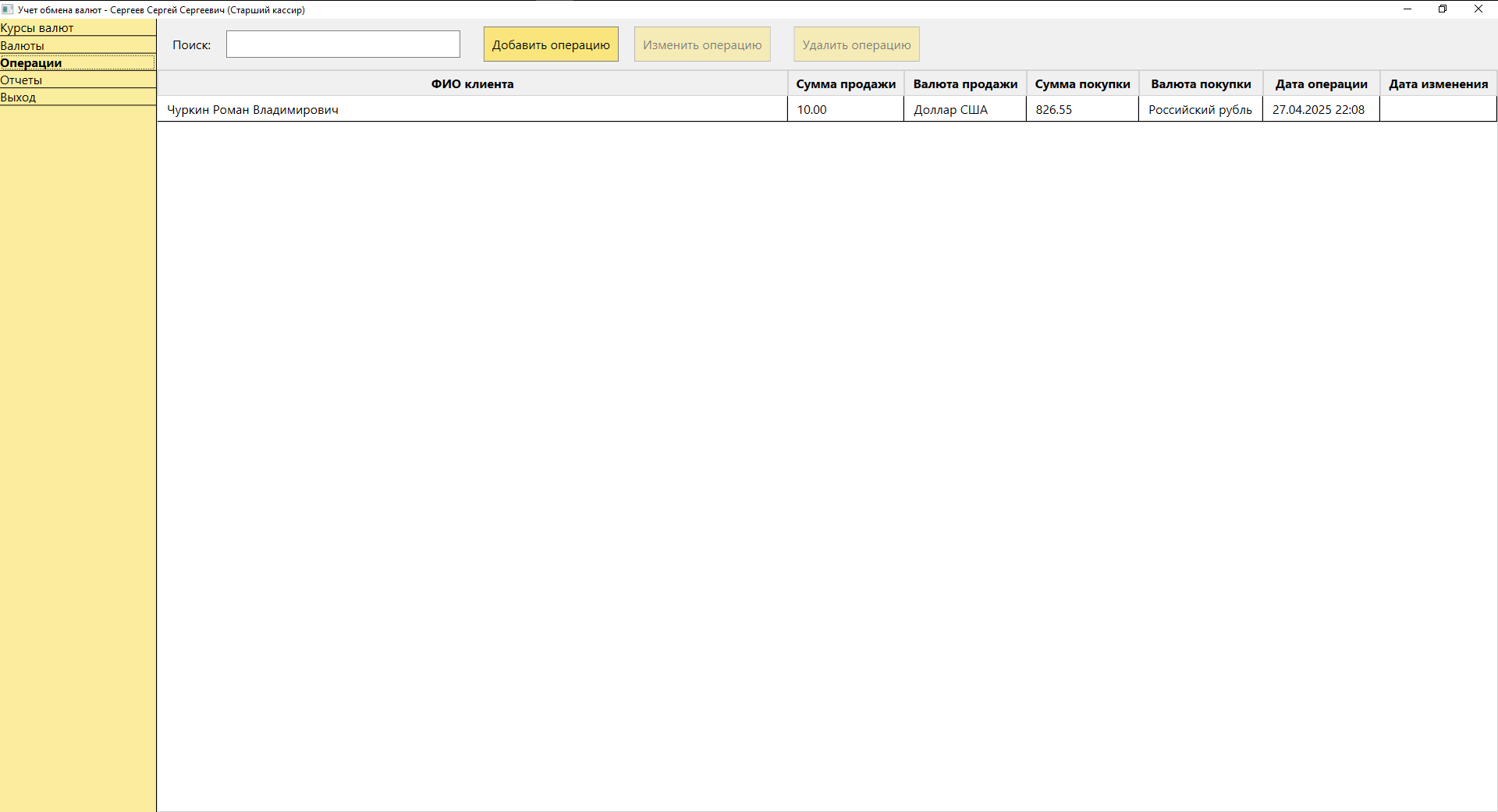
}

В модуле реализовано окно для управления валютами. При загрузке страницы данные о всех доступных валютах подгружаются из базы данных и отображаются в таблице. Пользователь может фильтровать валюты по имени или коду через специальное поле поиска. Таблица поддерживает сортировку данных по любому из столбцов (название, остаток, код, символ).

В верхней части страницы расположена панель с кнопкой "Изменить остаток", которая становится активной только при выборе валюты в таблице и доступна исключительно старшему кассиру. При клике на эту кнопку открывается форма для редактирования остатка выбранной валюты.

Форма редактирования содержит поля для отображения текущего имени валюты, текущего остатка и нового остатка. Пользователь может ввести корректное положительное число в поле для нового остатка и сохранить изменения, которые немедленно записываются в базу данных. После сохранения пользователь автоматически возвращается на страницу со списком валют.

### **3.2.4 Разработка модуля «Операции»**



*Рис. 3.2.4.1 Окно «Операции»*

Методы удаления операции и обработчик кнопки изменения операции окна «Операции»:

private void DeleteSelectedOperation()

{

if (\_selectedOperation == null) // Проверка выбора

{

MessageBox.Show("Выберите операцию для удаления", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Подтверждение удаления

var result = MessageBox.Show(

"Вы уверены, что хотите удалить эту операцию?",

"Подтверждение удаления",

MessageBoxButton.YesNo,

MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

using (var context = new CurrencyExchangeAccountingEntities())

{

// Поиск операции для удаления

var operationToDelete = context.Operations

.FirstOrDefault(o => o.Operation\_ID == \_selectedOperation.Operation\_ID);

if (operationToDelete != null)

{

// Получение валют

var currencyFrom = context.Currencies.Find(operationToDelete.Currency\_ID\_From);

var currencyTo = context.Currencies.Find(operationToDelete.Currency\_ID\_To);

// Корректировка остатков

if (currencyFrom != null)

currencyFrom.Remaining\_Amount += operationToDelete.Amount\_From;

if (currencyTo != null)

currencyTo.Remaining\_Amount -= operationToDelete.Amount\_To;

// Подсчет операций клиента

var clientOperationsCount = context.Operations

.Count(o => o.Client\_ID == \_selectedOperation.Client\_ID);

var client = context.Clients.Find(\_selectedOperation.Client\_ID);

context.Operations.Remove(operationToDelete); // Удаление операции

// Удаление клиента, если это его последняя операция

if (clientOperationsCount == 1 && client != null)

{

context.Clients.Remove(client);

}

context.SaveChanges(); // Сохранение изменений

App.RaiseCurrenciesUpdated(); // Обновление данных

LoadOperationsData(); // Перезагрузка данных

MessageBox.Show("Операция успешно удалена", "Успех",

MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

// Метод для обработки нажатия кнопки изменения

private void EditBtn\_Click(object sender, RoutedEventArgs e)

{

if (\_selectedOperation == null)

{

MessageBox.Show("Выберите операцию для изменения", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Асинхронный переход на страницу редактирования

Dispatcher.BeginInvoke(new Action(() =>

{

if (NavigationService != null)

{

NavigationService.Navigate(new EditOperationsPageSenior(\_currentUser, \_selectedOperation));

}

else

{

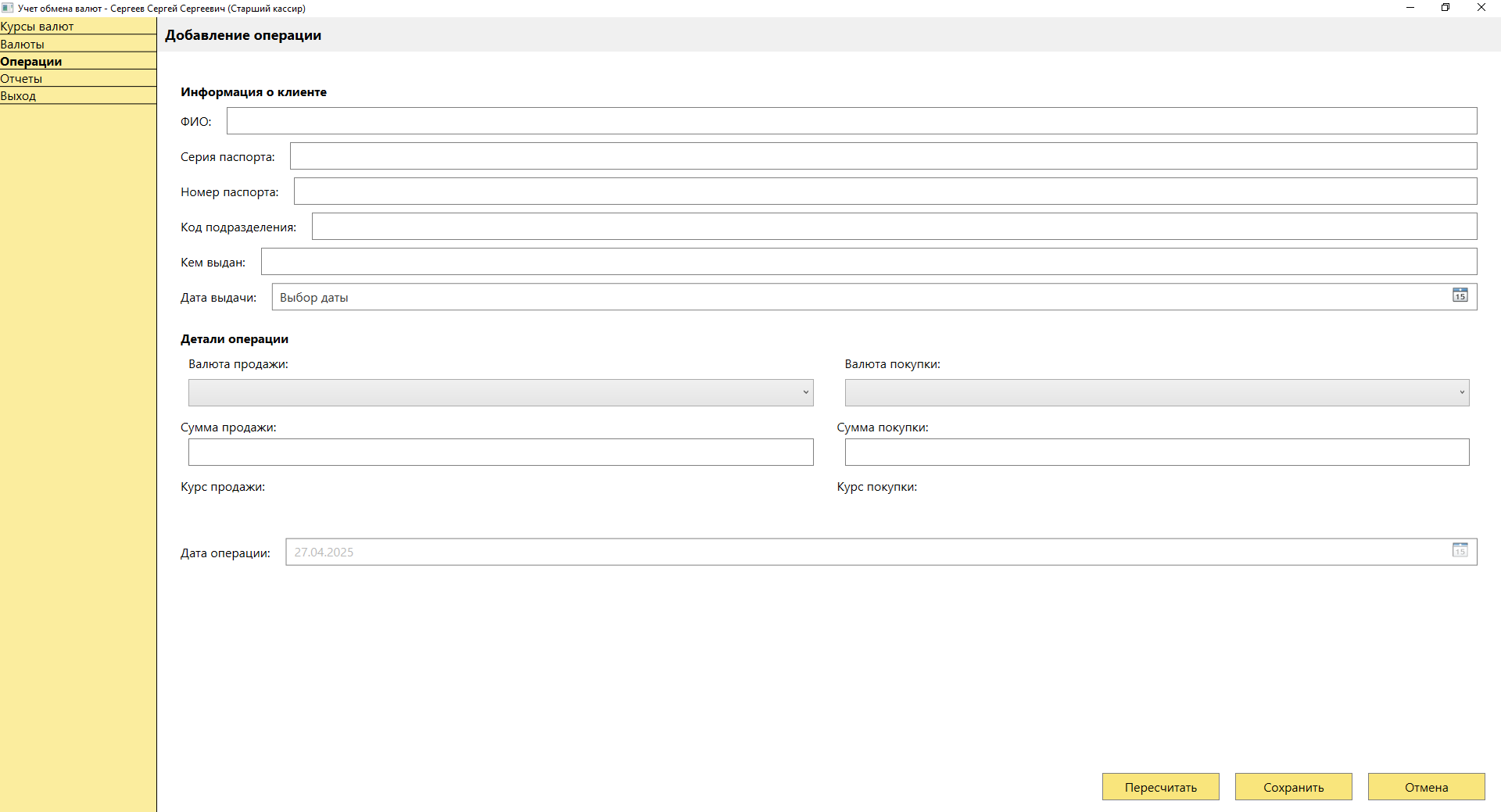
MessageBox.Show("Ошибка навигации. Попробуйте ещё раз.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

}

}));

}



*Рис. 3.2.4.2 Окно «Добавление операции»*

Методы расчета суммы обмена, загрузки данных валют и их курсов из базы данных окна «Добавление операции»:

// Загрузка данных валют

private void LoadData()

{

try

{

using (var context = new CurrencyExchangeAccountingEntities())

{

// Загрузка валют с курсами

var currencies = context.Currencies

.Include("ExchangeRates") // Включение курсов валют

.ToList();

// Настройка ComboBox для валют

CurrencyFromComboBox.ItemsSource = currencies;

CurrencyToComboBox.ItemsSource = currencies;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Обработчик кнопки расчета операции

private void CalculateButton\_Click(object sender, RoutedEventArgs e)

{

// Проверка заполнения обязательных полей

if (CurrencyFromComboBox.SelectedItem == null ||

CurrencyToComboBox.SelectedItem == null ||

string.IsNullOrWhiteSpace(AmountFromTextBox.Text))

{

MessageBox.Show("Заполните все поля для расчета", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Проверка корректности суммы

if (!decimal.TryParse(AmountFromTextBox.Text, out decimal amountFrom) || amountFrom <= 0)

{

MessageBox.Show("Введите корректную сумму для продажи", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Получение выбранных валют

var currencyFrom = (Currencies)CurrencyFromComboBox.SelectedItem;

var currencyTo = (Currencies)CurrencyToComboBox.SelectedItem;

// Отображение курсов валют

BuyRateTextBlock.Text = currencyFrom.ExchangeRates.Buy\_Rate.ToString("N4");

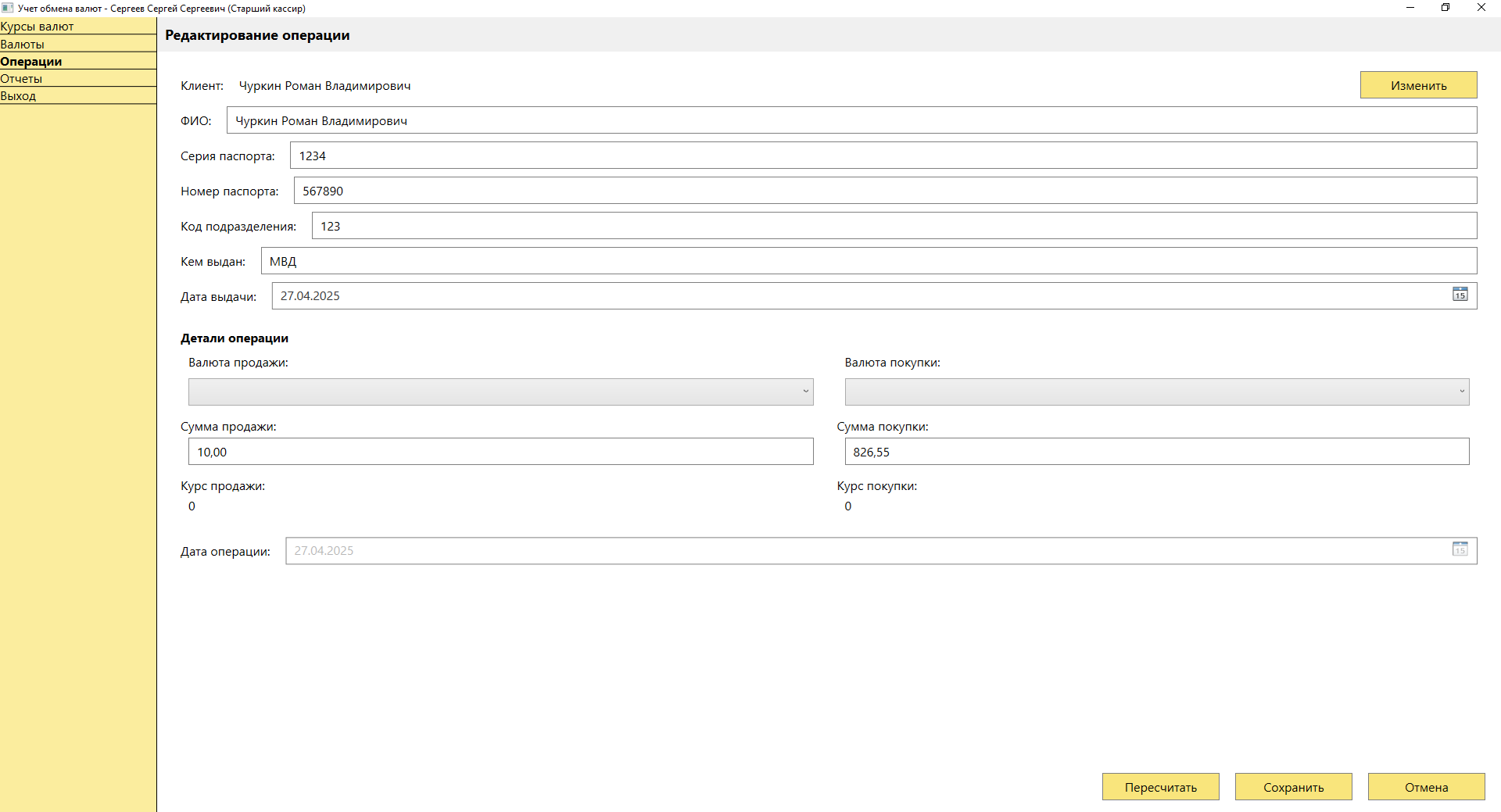
SellRateTextBlock.Text = currencyTo.ExchangeRates.Sell\_Rate.ToString("N4");

// Расчет суммы покупки

decimal amountTo = amountFrom \* currencyFrom.ExchangeRates.Buy\_Rate / currencyTo.ExchangeRates.Sell\_Rate;

AmountToTextBox.Text = amountTo.ToString("N2"); // Отображение результата

}



*Рис. 3.2.4.3 Окно «Редактирование операции»*

Методы расчета суммы обмена, загрузки данных валют и их курсов из базы данных окна «Редактирование операции»:

private void LoadData()

{

if (\_operationToEdit == null) // Проверка наличия операции

{

MessageBox.Show("Не удалось загрузить данные операции", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

using (var context = new CurrencyExchangeAccountingEntities()) // Новый контекст БД

{

// Установка имени клиента

ClientNameTextBlock.Text = \_operationToEdit.Clients.Client\_Full\_Name;

// Загрузка валют с курсами

var currencies = context.Currencies

.Include("ExchangeRates") // Включение курсов валют

.ToList();

// Настройка ComboBox для валют

CurrencyFromComboBox.ItemsSource = currencies;

CurrencyToComboBox.ItemsSource = currencies;

// Установка выбранных валют

CurrencyFromComboBox.SelectedValue = \_operationToEdit.Currency\_ID\_From;

CurrencyToComboBox.SelectedValue = \_operationToEdit.Currency\_ID\_To;

// Установка суммы продажи

AmountFromTextBox.Text = \_operationToEdit.Amount\_From.ToString("N2");

// Получение выбранных валют

var currencyFrom = (Currencies)CurrencyFromComboBox.SelectedItem;

var currencyTo = (Currencies)CurrencyToComboBox.SelectedItem;

// Установка курсов валют

BuyRateTextBlock.Text = currencyFrom?.ExchangeRates?.Buy\_Rate.ToString("N4") ?? "0";

SellRateTextBlock.Text = currencyTo?.ExchangeRates?.Sell\_Rate.ToString("N4") ?? "0";

// Установка суммы покупки

AmountToTextBox.Text = \_operationToEdit.Amount\_To.ToString("N2");

// Установка даты операции

OperationDatePicker.SelectedDate = \_operationToEdit.Operation\_Date;

}

}

catch (Exception ex) // Обработка ошибок

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void CalculateButton\_Click(object sender, RoutedEventArgs e)

{

// Проверка заполнения полей

if (CurrencyFromComboBox.SelectedItem == null ||

CurrencyToComboBox.SelectedItem == null ||

string.IsNullOrWhiteSpace(AmountFromTextBox.Text))

{

MessageBox.Show("Заполните все поля для расчета", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Проверка корректности суммы

if (!decimal.TryParse(AmountFromTextBox.Text, out decimal amountFrom) || amountFrom <= 0)

{

MessageBox.Show("Введите корректную сумму для продажи", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

// Получение выбранных валют

var currencyFrom = (Currencies)CurrencyFromComboBox.SelectedItem;

var currencyTo = (Currencies)CurrencyToComboBox.SelectedItem;

// Отображение курсов

BuyRateTextBlock.Text = currencyFrom.ExchangeRates.Buy\_Rate.ToString("N4");

SellRateTextBlock.Text = currencyTo.ExchangeRates.Sell\_Rate.ToString("N4");

// Расчет суммы покупки

decimal amountTo = amountFrom \* currencyFrom.ExchangeRates.Buy\_Rate / currencyTo.ExchangeRates.Sell\_Rate;

AmountToTextBox.Text = amountTo.ToString("N2"); // Отображение результата

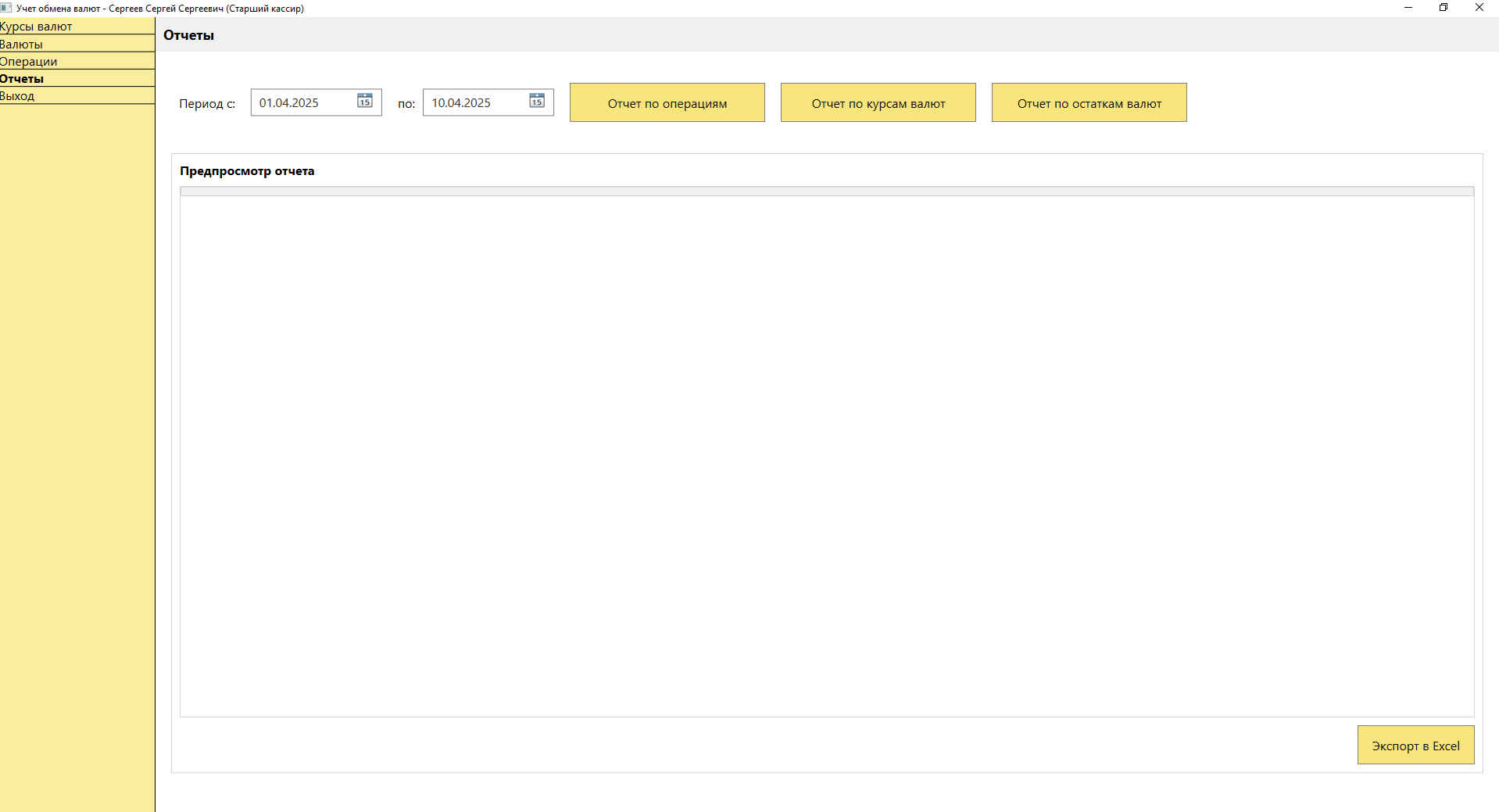
}

В модуле реализовано окно для управления операциями. При загрузке страницы отображается таблица с данными о всех операциях, включая ФИО клиента, сумму продажи, валюту продажи, сумму покупки, валюту покупки, дату операции и дату последнего обновления. Пользователь может фильтровать операции по ФИО клиента, валюте продажи или покупки через специальное поле поиска.

В верхней части страницы расположена панель с кнопкой "Добавить операцию. При клике на эту кнопку открывается форма для добавления новой операции.

Кнопка "Удалить операцию" доступна только старшему кассиру. При этом кнопка становится активной только если выбрана хотя бы одна операция в таблице. После успешного удаления операции пользователь автоматически возвращается на страницу со списком операций.

### **3.2.5 Разработка модуля «Отчеты»**



*Рис 3.2.5.1 Окно «Отчеты»*

Метод для создание отчета в Excel окна «Отчеты»:

// Метод для экспорта отчета в Excel

private void ExportToExcel()

{

// Проверка наличия данных для экспорта

if (\_currentReportData == null || \_currentReportData.Rows.Count == 0)

{

MessageBox.Show("Нет данных для экспорта", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

Application excelApp = null;

Workbook workbook = null;

try

{

// Создание экземпляра Excel

excelApp = new Application();

excelApp.Visible = true;

// Создание нового рабочего листа

workbook = excelApp.Workbooks.Add();

Worksheet worksheet = (Worksheet)workbook.ActiveSheet;

// 1. Запись заголовков

for (int i = 0; i < \_currentReportData.Columns.Count; i++)

{

worksheet.Cells[1, i + 1] = \_currentReportData.Columns[i].ColumnName;

}

// 2. Запись данных

for (int i = 0; i < \_currentReportData.Rows.Count; i++)

{

for (int j = 0; j < \_currentReportData.Columns.Count; j++)

{

worksheet.Cells[i + 2, j + 1] = \_currentReportData.Rows[i][j];

}

}

// 3. Форматирование заголовков

Range headerRange = worksheet.Range[

worksheet.Cells[1, 1],

worksheet.Cells[1, \_currentReportData.Columns.Count]];

headerRange.Font.Bold = true;

headerRange.Interior.Color = XlRgbColor.rgbLightGray;

headerRange.HorizontalAlignment = XlHAlign.xlHAlignCenter;

// 4. Добавление границ для всей таблицы

Range dataRange = worksheet.Range[

worksheet.Cells[1, 1],

worksheet.Cells[\_currentReportData.Rows.Count + 1, \_currentReportData.Columns.Count]];

dataRange.Borders.LineStyle = XlLineStyle.xlContinuous;

dataRange.Borders.Weight = XlBorderWeight.xlThin;

// 5. Автоподбор ширины столбцов

worksheet.Columns.AutoFit();

// 6. Отключение автофильтров

try

{

worksheet.AutoFilterMode = false;

}

catch

{

// Игнорирование ошибок при отключении фильтров

}

// Сообщение об успешном экспорте

MessageBox.Show("Отчет успешно экспортирован в Excel",

"Успех",

MessageBoxButton.OK,

MessageBoxImage.Information);

}

catch (Exception ex)

{

// Обработка ошибок при экспорте

MessageBox.Show($"Ошибка при экспорте в Excel: {ex.Message}",

"Ошибка",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

finally

{

// Освобождение ресурсов

if (workbook != null)

Marshal.ReleaseComObject(workbook);

if (excelApp != null)

Marshal.ReleaseComObject(excelApp);

}

}

В модуле реализовано окно для просмотра и генерации отчетов, доступное только старшему кассиру. При загрузке страницы отображается интерфейс с полями выбора дат и кнопками для генерации различных отчетов. Пользователь может выбрать период для отчета. После выбора диапазона дат доступны кнопки для генерации отчетов по операциям, курсам валют и остаткам валют.

В нижней части страницы находится таблица для отображения данных отчета. При генерации отчета данные загружаются из базы данных и отображаются в таблице. Таблица поддерживает автогенерацию столбцов и форматирование данных.

При нажатии на кнопку "Экспорт в Excel" отчет преобразуется в формат, подходящий для Microsoft Excel, с форматированием заголовков, границ и автоподбором ширины столбцов.

# **ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы была разработана автоматизированная система "Учет обмена валют", предназначенная для оптимизации работы финансовых организаций. Система обеспечивает автоматизацию ключевых процессов: обработку курсов валют, фиксацию операций и формирование отчетности. Особое внимание уделено удобному интерфейсу для кассиров, а также надежной системе безопасности данных.

Практическая ценность системы заключается в значительном сокращении времени обработки операций и минимизации ошибок. Автоматизированный учет позволяет повысить точность финансовой отчетности и снизить операционные риски. Реализованные механизмы авторизации и резервного копирования обеспечивают защиту конфиденциальной информации.

Разработанное решение полностью соответствует современным требованиям к финансовым системам. Тестирование подтвердило работоспособность всех модулей и готовность к внедрению. Система обладает потенциалом для дальнейшего развития и адаптации под конкретные задачи банков и обменных пунктов.

Репозиторий GitHub - <https://github.com/tqarrx/Course-project-CEA->

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Microsoft Docs. Общие сведения о .NET. [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/core/introduction?WT.mc_id=dotnet-35129-website>.

2. Microsoft Docs. Обзор C#. [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/overview>.

3. Microsoft Docs. Введение в Windows Presentation Foundation (WPF). [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/introduction-to-wpf?view=netframeworkdesktop-4.8>.

4. Microsoft Docs. Общие сведения о SQL Server. [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/sql-server/what-is-sql-server?view=sql-server-ver16>.

5. Microsoft Docs. SQL Server Management Studio (SSMS). [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/ssms/sql-server-management-studio-ssms?view=sql-server-2017>.

6. Microsoft Docs. Общие сведения об Entity Framework. [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/overview>.

7. Lucidchart. ERD диаграмма: руководство по созданию. [Электронный ресурс] – Режим доступа: <https://www.lucidchart.com/pages/ru/erd-diagram>.