**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc169590019)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 6](#_Toc169590020)

[1.1. Описание предметной области 6](#_Toc169590021)

[1.2. Анализ 10](#_Toc169590022)

[1.3. Аналитическое описание выявленных достоинств и недостатков 12](#_Toc169590023)

[1.4. Используемый язык программирования С# 14](#_Toc169590024)

[1.5. Общая постановка задачи 15](#_Toc169590025)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 17](#_Toc169590026)

[2.1. Концептуальное проектирование программы 17](#_Toc169590027)

[2.2. Структура программного продукта 18](#_Toc169590028)

[2.3. Руководство пользователю 20](#_Toc169590029)

[2.4. Руководство программисту 28](#_Toc169590030)

[ЗАКЛЮЧЕНИЕ 33](#_Toc169590031)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc169590032)

# ВВЕДЕНИЕ

В последние годы банковские приложения стали неотъемлемой частью повседневной жизни, обеспечивая удобство и безопасность финансовых операций, что делает изучение и разработку таких систем особенно важной темой для всей индустрии информационных технологий.

Актуальность выбранной темы курсовой работы заключается в неразрывной связи современного общества с финансовыми операциями, где банковские системы играют ключевую роль в управлении денежными средствами своих клиентов. С развитием технологий растет потребность в удобных и надежных банковских приложениях, которые обеспечивают простой доступ к управлению своими финансами. В условиях цифровой трансформации банковского сектора важно понимать, как создавать и развивать такие приложения.

Основная цель курсовой работы заключается в изучении языка программирования C# и проектировании архитектуры клиент-серверного приложения на примере банковской системы. Достижение этой цели позволит не только углубить знания в области программирования, но и понять принципы функционирования банковских систем.

**Для достижения цели курсовой работы были поставлены следующие задачи:**

1. Изучение теоретических аспектов банковского дела.
2. Анализ существующих банковских приложений для выявления ключевых функций приложения.
3. Создание модели пользовательского интерфейса (UI) и базы данных для хранения информации о клиентах и транзакциях.
4. Разработка серверной части банковского приложения на ASP.NET Core.
5. Разработка клиентской части банковского приложения на .NET MAUI.
6. Тестирование прототипа с использованием различных сценариев для выявления и устранения ошибок, а также для улучшения производительности и удобства использования приложения.
7. Сравнение разработанного приложения с существующими решениями на рынке по функциональности, удобству использования и безопасности, а также оценка его преимуществ и недостатков.

**Объектом** исследования является банковская информационная система.

**Предметом** исследования выступает разработка приложения, имитирующего банковскую систему.

**Методы исследования:**

* **Анализ и синтез**: Анализ существующих банковских приложений для выявления ключевых функций и синтез результатов анализа для определения необходимых функций и требований к приложению.
* **Моделирование**: Проектирование архитектуры модели пользовательского интерфейса (UI) и базы данных, разработка прототипа приложения.
* **Эксперимент**: Тестирование прототипа с использованием различных сценариев использования, выявление и устранение ошибок, улучшение производительности и удобства использования приложения.
* **Сравнение**: Сравнение разработанного приложения с существующими решениями на рынке по функциональности, удобству использования и безопасности, оценка его преимуществ и недостатков.

В первой главе курсовой работы рассматриваются теоретические аспекты банковских информационных систем, анализ существующих решений, определение ключевых функций и требований к приложению. Во второй главе курсовой работы формулируется концептуальное проектирование, структура программного продукта, предлагается руководство пользователю и разработчику. В приложении представлен листинг разработанного прототипа банковского приложения.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## 1.1. Описание **предметной** области

Под банковской системой (БС) понимается совокупность процедур, людей и ресурсов, которая собирает, обрабатывает и распространяет финансовую информацию в той организации, к которой она относится. У банковской системы, как и у любой динамической системы, есть три ключевые функции:

1. Ввод (получение и накопление данных о клиентах, счетах, картах и транзакциях).
2. Обработка (преобразование входных данных в полезные информационные продукты, такие как отчеты, выписки и аналитические данные).
3. Вывод (предоставление результатов обработки пользователям, включая клиентов и сотрудников банка).

Автоматизированная банковская система (АБС) использует средства вычислительной техники, программное обеспечение, телекоммуникации и прочие разновидности информационных технологий (ИТ) для преобразования имеющихся данных в информационные продукты. Конечным пользователем является любой, кто использует АБС или информацию, производимую ею, включая клиентов банка и сотрудников.

АБС применяется для предоставления информации, помогающей эффективно производить финансовые операции и осуществлять управление банковской организацией. Автоматизированная банковская система обеспечивает управление счетами, обработку транзакций, обслуживание клиентов и обеспечение безопасности.

Банк — это финансовое учреждение, организующее сбор, хранение и управление денежными средствами и другими финансовыми активами. Банки систематически занимаются сбором, хранением и выдачей денежных средств, а также предоставлением различных финансовых услуг. Они являются важнейшими элементами экономической инфраструктуры и основными институтами для накопления и распределения капитала.

Основными направлениями работы любого банка являются:

* обслуживание клиентов;
* управление счетами и картами;
* проведение транзакций и платежей.

Обслуживание клиентов банка включает в себя предоставление консультаций, открытие и закрытие счетов, выпуск и обслуживание банковских карт, а также предоставление информации о текущем состоянии счетов и проведенных транзакциях. От качества обслуживания клиентов во многом зависит уровень их удовлетворенности и доверия к банку.

Управление счетами включает открытие, закрытие, учет и контроль счетов клиентов. Это процесс, который обеспечивает безопасное и эффективное хранение и использование денежных средств клиентов. Счета могут быть открыты в различных валютах и использоваться для различных целей, включая накопление сбережений и проведение расчетов.

Процесс выпуска и обслуживания банковских карт также является важной частью работы банка. Клиенты могут использовать карты для снятия наличных, проведения платежей и других финансовых операций.

Банковская деятельность также связана с учетом большого количества операций и обеспечением безопасности финансовых активов клиентов. Множество клиентов и разнообразие финансовых операций требуют от банковского персонала высокой компетентности и оперативности в выполнении задач.

Процесс организации финансовых активов в банке является ключевым для обеспечения удобства клиентов и эффективного управления активами. Правильная организация счетов и карт облегчает клиентам проведение финансовых операций, а банковским специалистам — быстрое выполнение запросов клиентов и поддержание сохранности финансовых средств.

Обслуживание клиентов в банке осуществляется через предоставление доступа к счетам и картам, помощь клиентам в выборе финансовых продуктов и оперативное решение финансовых вопросов. Клиенты для получения доступа к своим счетам и карточным операциям авторизуются, то есть предоставляют идентификационные данные.

Банковская деятельность включает в себя ведение подробной базы данных, которая хранит следующие данные о пользователях, счетах, картах и транзакциях:

О пользователе:

* Имя
* Фамилия
* Почта
* Пароль (зашифрованный)
* Дата регистрации
* Пользовательский ID
* Роль (Клиент\Админ)

О счетах:

* Номер счета
* ID клиента, кому принадлежит счет
* Валюта счета
* Баланс
* Дата открытия
* Дата закрытия (если счет закрыт)
* Активность счета
* Привязанная карта к счету

О карте:

* Номер карты
* Клиентский ID, которому принадлежит карта
* Тип карты (Visa, MasterCard и т.д.)
* Дата выпуска
* Дата окончания обслуживания
* Пинкод (зашифрованный)
* CVC код (зашифрованный)
* Счет, привязанный к карте (главный счет оплаты)

О транзакциях:

* Уникальный GUID транзакции
* Клиентский ID отправителя
* Клиентский ID получателя
* Счет отправления
* Счет получателя
* Цель транзакции
* Размер транзакции

Эти данные составляют основу информационной системы банка, обеспечивающей автоматизированную обработку финансовых операций, учет финансовых активов клиентов и обеспечение безопасности их персональных данных. В банке существует система учета и контроля за счетами и картами клиентов. Это включает регистрацию новых счетов и карт, ведение учета выданных кредитов и счетов, а также контроль за сроками действия карт.

## 1.2. Анализ

Автоматизированная банковская система (АБС) — незаменимый инструмент для эффективного управления клиентскими данными, счетами, картами и транзакциями в современных банках. Эта система достаточно сложна в настройке и эксплуатации, поэтому её выбор нередко обусловлен не только потребностями, но и техническими возможностями банка. Основная цель автоматизированной банковской информационной системы — хранение, обеспечение эффективного поиска и передачи информации по соответствующим запросам для наиболее полного удовлетворения потребностей клиентов и обеспечения высокого уровня сервиса.

По мнению многих руководителей банков и специалистов по финансовым технологиям, идеальной АБС не существует, а переход с одной системы на другую слишком затратно, чтобы менять поставщиков программного продукта. Тем не менее, несмотря на сложности миграции, все ведущие системы автоматизации поддерживают дух конкурентной борьбы, не останавливают своего развития и предлагают, при внешнем сходстве различных программных решений, уникальные особенности и функциональные возможности. Рассмотрим примеры существующих разработок автоматизированных банковских систем:

**Сбербанк**

Сбербанк — крупнейший банк России и один из ведущих финансовых институтов в мире. Программное обеспечение банка представляет собой сложную систему, обеспечивающую высокую степень безопасности и удобства для пользователей. Рассмотрим клиентскую часть банковской системы Сбербанка.

**Клиентская часть:**

* Платформа: iOS, Android.
* Требования к ресурсам: средние.
* Поддержка облачных технологий: да.

Мобильное приложение Сбербанка предоставляет широкий спектр услуг, позволяя клиентам выполнять множество финансовых операций прямо со своих мобильных устройств. Также в разработке Сбербанка следует выделить:

**Удобный интерфейс:** Приложение разработано с учетом современных стандартов UX/UI, что делает его использование интуитивно понятным и удобным для пользователей всех возрастов.

**Многофункциональность:** Клиенты могут проверять баланс, переводить деньги, оплачивать услуги, открывать депозиты, оформлять кредиты и многое другое.

**Высокий уровень безопасности:** Приложение поддерживает двухфакторную аутентификацию, а также шифрование данных, что обеспечивает надежную защиту пользовательской информации.

**Интеграция с другими сервисами:** Приложение интегрировано с различными сервисами и платформами, что позволяет пользователям, например, оплачивать покупки через Apple Pay или Google Pay.

**Агропромбанк**

Агропромбанк – региональный банк Приднестровской Молдавской Республики, оказывающий финансовые услуги подавляющему большинству граждан республики. Мобильное приложение банка обладает высокой степенью защиты, а также широким функционалом, как для обычных физических лиц, клиентов банка, так и для юридических лиц с возможность интеграции их предприятий в банковскую систему Агропромбанка. Таким образом в разработке Агропромбанка следует выделить:

**Высокую степень интеграции:** Приложение Агропромбанка, позволяет производить оплату коммунальных платежей, покупку билетов на культмассовые мероприятия Республики, перевод между картами, выпущенных различными картами, благодаря системе платежей Клевер.

**Понятный интерфейс:** Рассматриваемое приложение не обладает уникальными дизайнерскими решениями, но строгость и четкость делает его использование простым и удобным для пользователей среднего возраста.

**Высокая степень защиты:** Приложение поддерживает систему сеансов с минимальным временем активности, благодаря этому, злоумышленники с меньшей вероятностью авторизуются, используя ваш ключ сеанса. Так же доступ к сервису банка, пароль и логин, клиенты получают в отделениях банка, что также повышает надежность банковской системы Агропромбанка.

Каждая из представленных банковских систем имеет свои особенности, достоинства и недостатки. Так, Сбербанк обладает удобным и современным интерфейс, в то время как Агропромбанк делает ставку на консервативные решения в плане дизайна, но стоит отметить, что оба сервиса обладает высокой степенью интеграции с сервисами и услугами, такими как: коммунальные услуги, интернет-покупки, приобретение билетов на культурные мероприятия в регионе.

## 1.3. Аналитическое описание выявленных достоинств и недостатков

Аналитическое описание достоинств и недостатков информационной системы банка:

**Достоинства:**

**Автоматизация банковских операций:** Использование информационной системы позволяет банку автоматизировать процессы учета транзакций, клиентских счетов и операций с картами. Это значительно уменьшает риск ошибок, связанных с ручным ведением учета, и повышает операционную эффективность банка.

**Улучшенное обслуживание клиентов:** Система предоставляет клиентам удобный доступ к банковским услугам через различные каналы: мобильное приложение, интернет-банкинг и терминалы самообслуживания. Это обеспечивает высокую доступность услуг и удовлетворение клиентских потребностей.

**Аналитические возможности:** Информационная система позволяет банку осуществлять мониторинг операций, таких как количество сделок в день или объемы транзакций по различным валютным парам. Это обеспечивает оперативное реагирование на изменения на рынке и потребности клиентов.

**Недостатки:**

**Сложность и длительность обновлений:** Любое крупное обновление банковской информационной системы — это долгий и сложный процесс, который требует тщательного планирования и учета всех данных клиентов. Недостаточная внимательность к деталям или ошибки в процессе миграции могут стать причиной финансовых потерь и ущерба для репутации банка.

**Риск несанкционированного доступа и безопасность:** Одним из критически важных аспектов является защита информационной системы от несанкционированного доступа и киберугроз. Недостаточная защита данных и клиентской информации может привести к утечкам конфиденциальной информации, финансовым мошенничествам и нарушению репутации банка. Потери данных или доступ к клиентским счетам могут нанести серьезный ущерб как финансовым ресурсам банка, так и его имиджу перед клиентами и общественностью.

**Зависимость от технической поддержки:** Использование сложных банковских систем требует постоянной технической поддержки и обновлений. Необходимость в квалифицированных IT-специалистах и затраты на поддержание высокого уровня безопасности и производительности могут быть значительными. В случае недостаточной поддержки или задержек в обновлениях возрастает риск технических сбоев или уязвимостей, что может негативно отразиться на операционной деятельности банка.

## 1.4. Используемый язык программирования С#

Для разработки банковской системы был выбран С# («Си Шарп») — один из наиболее быстро растущих, востребованных и при этом «удобных» языков программирования. C# (C Sharp) разработан компанией Microsoft и является частью .NET Framework/.NET Core. Этот язык программирования сочетает в себе синтаксическую ясность и выразительность Си с современными возможностями программирования. Он был создан для упрощения разработки ПО на различных платформах, включая Windows, Linux и мобильные устройства. C# предлагает строгую типизацию, автоматическое управление памятью и поддержку современных технологий безопасности, что делает его предпочтительным выбором, и выделяет следующие аргументы в пользу данного языка программирования:

**Платформенная независимость и универсальность:** C# является частью .NET Framework/.NET Core, что обеспечивает высокую степень платформенной независимости. Банковская система, разработанная на C#, может быть легко адаптирована для работы как на серверах Windows, так и на Linux. Это позволяет обеспечить гибкость в выборе аппаратного и программного обеспечения, что критически важно для банковских систем, работающих на различных платформах.

**Надежность и управление памятью:** C# включает в себя механизмы автоматического управления памятью и сильную типизацию, что существенно снижает риск утечек памяти и повышает общую устойчивость кода. Это особенно важно для банковских систем, обрабатывающих большой объем данных от клиентов.

**Широкие возможности интеграции:** C# обеспечивает высокую степень интеграции с другими технологиями и сервисами благодаря мощным фреймворкам, таким как ASP.NET Core для веб-разработки и .NET MAUI для мобильных приложений. Это позволяет банкам создавать современные мультиплатформенные приложения, удовлетворяя потребности клиентов в доступе через различные устройства.

**Крупное сообщество и поддержка:** C# имеет активное сообщество разработчиков и обширную документацию, что обеспечивает доступ к широкому спектру знаний и поддержку. Это упрощает разработку, поддержку и масштабирование банковских систем, минимизируя риски и повышая эффективность проектов.

Так же стоит отметить недостатки языка программирования C#, которые следует учитывать при проектировании приложения:

**Производительность и скорость загрузки:** Приложения на C# требуют времени на инициализацию и адаптацию к конкретному аппаратному обеспечению, что может снизить скорость первоначальной загрузки и реакции интерфейсов.

**Уязвимость безопасности:** Существует риск декомпиляции кода, написанного на C#, что может угрожать безопасности и конфиденциальности данных, обрабатываемых банковскими системами.

## 1.5. Общая постановка задачи

Общая постановка задачи курсовой работы на тему «Имитация банковской системы» заключается в следующем:

Целью работы является разработка функциональной серверной и клиентской части банковской системы.

Задачи работы включают:

1. Анализ потребностей банка и выявление основных требований к учету данных и возможностей сервера.

2. Изучение теоретических аспектов банковских систем и обработки финансовых средств.

3. Выбор подходящего языка программирования.

4. Проектирование архитектуры банковской системы, включая базу данных и пользовательский интерфейс.

5. Разработка программного кода для реализации функциональности системы.

6. Тестирование и отладка разработанной банковской системы.

7. Оценка эффективности и функциональности системы, а также предложение возможных путей ее улучшения.

В результате выполнения работы ожидается создание функциональной банковской системы, способной предоставлять доступ и возможности управление денежными средствами клиентов банка.

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## **2.1. Концептуальное проектирование программы**

*Концептуальное проектирование* – это начальная стадия жизненного цикла проекта, на которой формируются основные идеи и принимаются ключевые решения по его реализации. Эти идеи и решения оказывают значительное влияние на весь проект, позволяя своевременно оценить его эффективность и риски. В рамках данного проекта, направленного на разработку имитации банковской системы, были определены следующие аспекты:

1. **Модель данных:**

* *Основные сущности:* Клиент, Счета, Транзакции, Карты.
* *Атрибуты сущностей:* Представлены в UML диаграмме сущностей.
* **Основные связи:**
  + *ID клиента:* присутствует в сущностях Счета (владелец счета), Транзакции (отправитель, получатель транзакции), Карта (владелец карты).
  + *Карты:* Карта обладает полем "Счет" (главный счет карты), что позволяет отслеживать баланс и осуществлять операции.
  + *Счет:* Поле "Карта" (привязанная карта) связывает счет с картой, обеспечивая возможность привязки множества счетов к одной карте, при этом на карте может быть выставлен лишь один главный счет.

1. **Пользовательский интерфейс:**

**Основные функции:**

* *Создание нового счета.*
* *Перевод средств.*
* *Обмен валют.*
* *Просмотр транзакций.*
* *Открытие карт.*
* *Поля для входа и регистрации.*
* *Формы:*
* *Регистрация клиента.*
* *Идентификация клиента (поле входа).*

1. **Хранение данных:**

* *База данных:* EntityFramework.Sqlite.
* *Таблицы:* Будут созданы таблицы Bills, Cards, Transactions, Users для хранения соответствующих сущностей и их атрибутов.

## 2.2. Структура программного продукта

Структура программного продукта имитации банковской системы состоит из двух представлений: Клиентской части (.NET MAUI) и Серверной части (ASP.NET Core с минимальными API).

Структура клиентского представления:

Страницы приложения:

1. MainPage. Страница, содержащая поля входа, а также переадресацию на регистрационную страницу. При условии авторизации пользователя ранее, данная форма автоматически переадресует на основную страницу.

2. RegisterPage. Страница, содержащая поля регистрации.

3. Container. Основная страница, содержащая вкладки:

3.1. MainProfile. Вкладка отображения последних транзакций по всем счетам, а так же основной информации о пользователе.

3.2. Operations. Вкладка, предоставляющая возможность управления денежными средствами на счетах и картах

3.3. Cards. Вкладка отображения карт клиента, с возможностью перехода на страницу взаимодействия с данной картой, а так же возможностью выпуска новой карты.

3.3.1. CardView. Страница для взаимодействия с картой, закрытия, получения секретной информации (CVC\PIN)

3.3.2. DialogueCard. Страница диалога с пользователем, предоставляющая возможность выпуска новой карты.

3.4. Bills. Вкладка отображения счетов клиента, с возможность перехода на страницу взаимодействия с данным счетом.

3.4.1. BillView. Страница для взаимодействия со счетом, получения выписки (истории транзакций)

Сущности, используемые приложением:

Сущности запросов на сервер Data-Transfer objects (DTO’s):

1. LoginDTO. Сущность, хранящая в себе строковые значения почты(логина) и пароля пользователя, для взаимодействия с серверной частью

2. RegisterDTO Сущность, хранящая в себе информацию о регистрации.

3. BillDTO. Сущность представления счета.

4. CardDTO. Сущность представления карты.

5. TransactionDTO. Сущность представления транзакции.

6. PaymentDTO. Сущность представления платежа.

Сущности отображения данных (ViewModel’s):

1. BillViewModel. Сущность, наследуемая от BillDTO, используемая для отображения данных в интерфейсе приложения.

2. CardViewModel. Сущность, наследуемая от CardDTO.

3. TransactionViewModel. Сущность, наследуемая от TransactionDTO.

Основные функции клиентского приложения:

1. Регистрация клиента. Пользователь может создать аккаунт для использования нашего приложения.

2. Создания нового счета. Пользователь может создать новый счет в любой из предложенных валют.

3. Создание новой карты. Пользователь может создать новую карту любого типа с выбранным сроком действия, оплатив комиссионное вознаграждение банку.

4. Перевести средства с любого своего счета на любой счет другого пользователя. Пользователь может переводить средства независимо от того одинакова ли валюта на счетах.

5. Обменять валюту. Пользователь может переводить средства между своими счетами, благодаря чему и происходит её обмен.

6. Получение секретных данных карты. Пользователь введя пароль, как каптчу, может получить доступ к данным своей карты, таким как: PIN, CVC.

7. Получение истории транзакций. Пользователь может просмотреть историю транзакций как по всем счетам сразу, так и в отдельности по каждому из данных.

8. Выйти из аккаунта. Пользователь может выйти из аккаунта, дабы войти в другой свой аккаунт не перезапуская приложения.

## 2.3. Руководство пользователю

Руководство пользователя предназначено для ознакомления с функциональными возможностями имитационной банковской системы. Оно поможет пользователям научиться эффективно использовать приложение для управления своими банковскими операциями.

Руководство пользователя обеспечивает полное представление о структуре интерфейса приложения и описывает все реализованные функции.

Инструкция пользователя

Запуск приложения

Чтобы запустить приложение, нужно дважды кликнуть левой кнопкой мыши (ЛКМ) по исполняемому файлу – файл с расширением .ехе. Данное действие показано на рисунке 1.

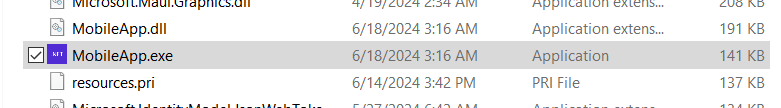


Рис. 1 Запуск приложения

Окно входа

После запуска приложения откроется главное окно ввода идентификационных данных, его можно увидеть на рисунке 2. На данном окне расположена кнопка входа и строка перехода на окно регистрации.

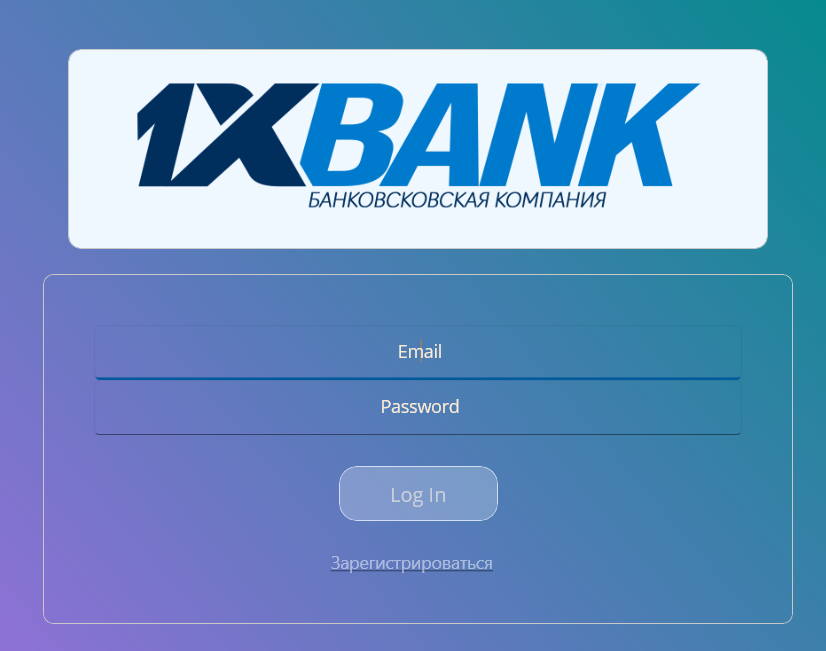


Рис. 2 Окно входа

Окно регистрации

Окно регистрации, изображенное на рисунке 3, не отличается от окна входа, здесь лишь увеличено кол-во полей ввода информации. Все-также тут есть кнопка регистрации и строка перехода на окно входа.

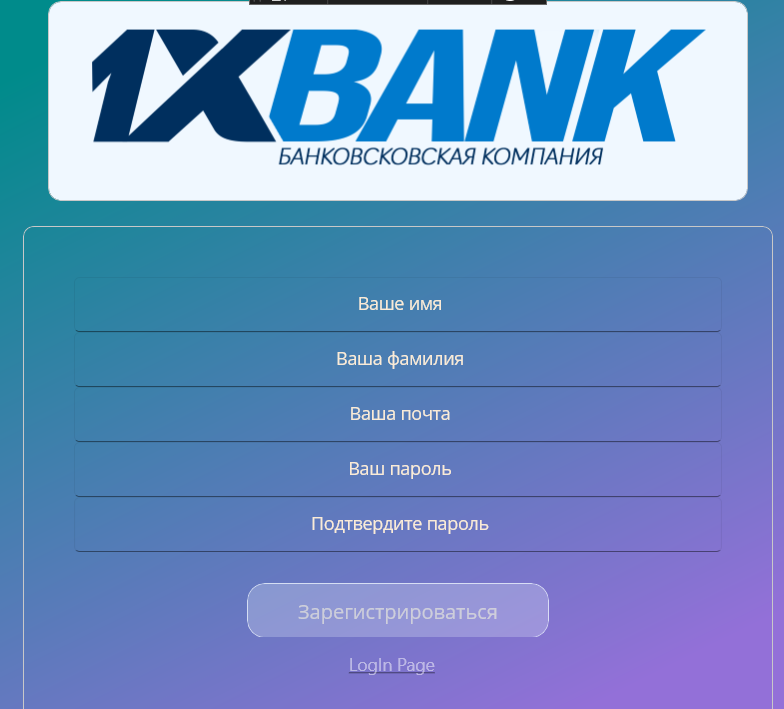


Рис. 3 Окно регистрации

Окно отображения карт пользователя

После успешной авторизации и загрузки приложения, пользователь перенаправляется на окно отображение его карт, окно продемонстрировано на рисунке 4. По нажатию ЛКМ на любую строку с номером карты открывается окно взаимодействия с данной картой. По нажатию ЛКП на кнопку «Выпустить новую карту», открывается окно диалога с пользователем с запросом информации для выпуска новой карты.

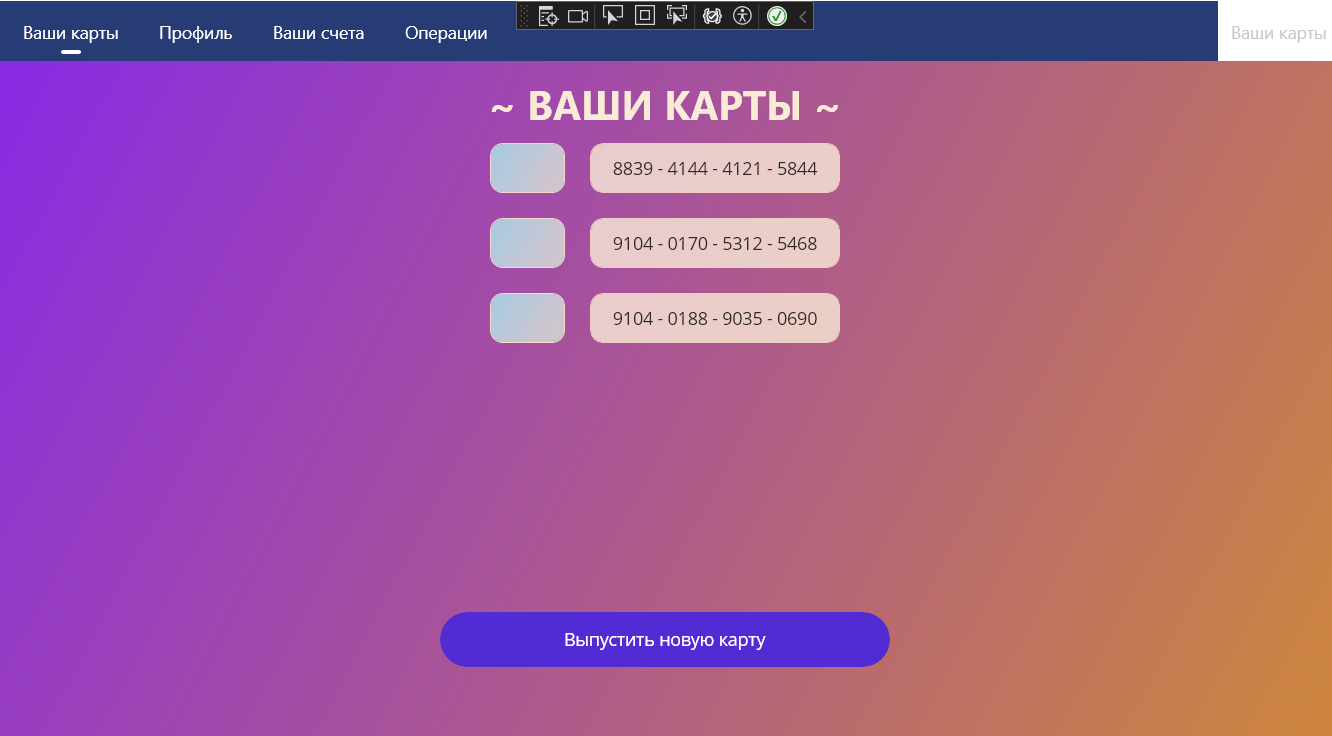


Рис. 4 Окно отображения карт пользователя

Окно взаимодействия с картой

По нажатию ЛКМ на строку с номером карты, открывается окно взаимодействия с картой, что видно на рисунке 5. Здесь отображается справочная информация о состоянии карты.

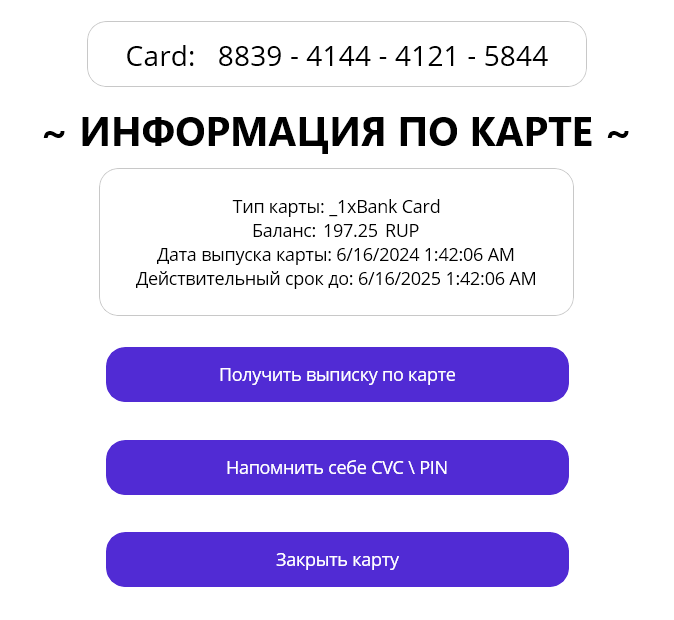


Рис. 5 Окно взаимодействия с картой

Так же здесь имеются 3 функциональные кнопки:

1. «Получить выписку по карте»:

Отображение транзакций связанных с данной картой.

2. «Напомнить себе CVC \ PIN»:

Запрос на получения секретных данных карты. Пользователю требуется ввести пароль от своего аккаунта, чтобы получить эти данные. После ввода корректных данных, пользователю будет высвечена запрашиваемая информация. Данные действия продемонстрированы на рисунках 6 и 7.

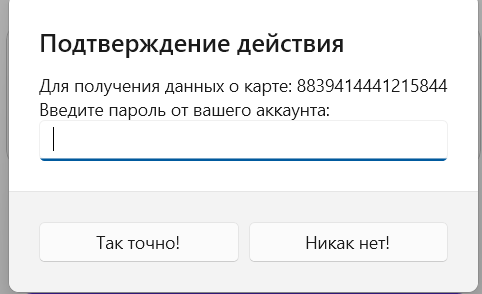


Рис. 6 Окно ввода данных

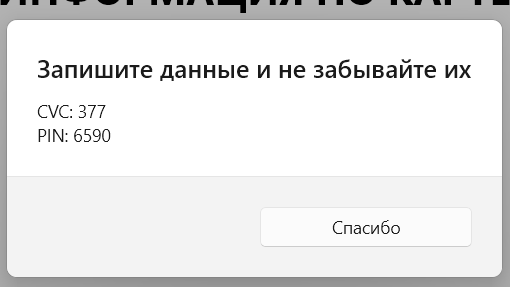


Рис. 7 Получение секретных данных о карте

3. «Закрыть карту»:

Запрос на закрытие карты, пользователю требуется ввести капчу (четырехзначное число). Данное действие показано на рисунке 8.

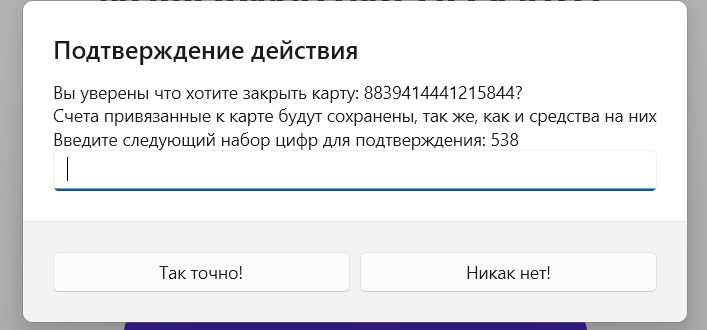


Рис. 8 Подтверждение закрытия карты

Окно выпуска новой карты

В окне отображения карт при нажатии на кнопку «Выпустить новую карту», появляется следующее окно, смотреть на рисунок 9, с соответствующими полями. В зависимости от срока действия карты будет зависеть её стоимость выпуска. Оплату можно производить в любой валюте, если на счету оплаты будет недостаточно средств, будет высвечено соответствующее этому сообщение

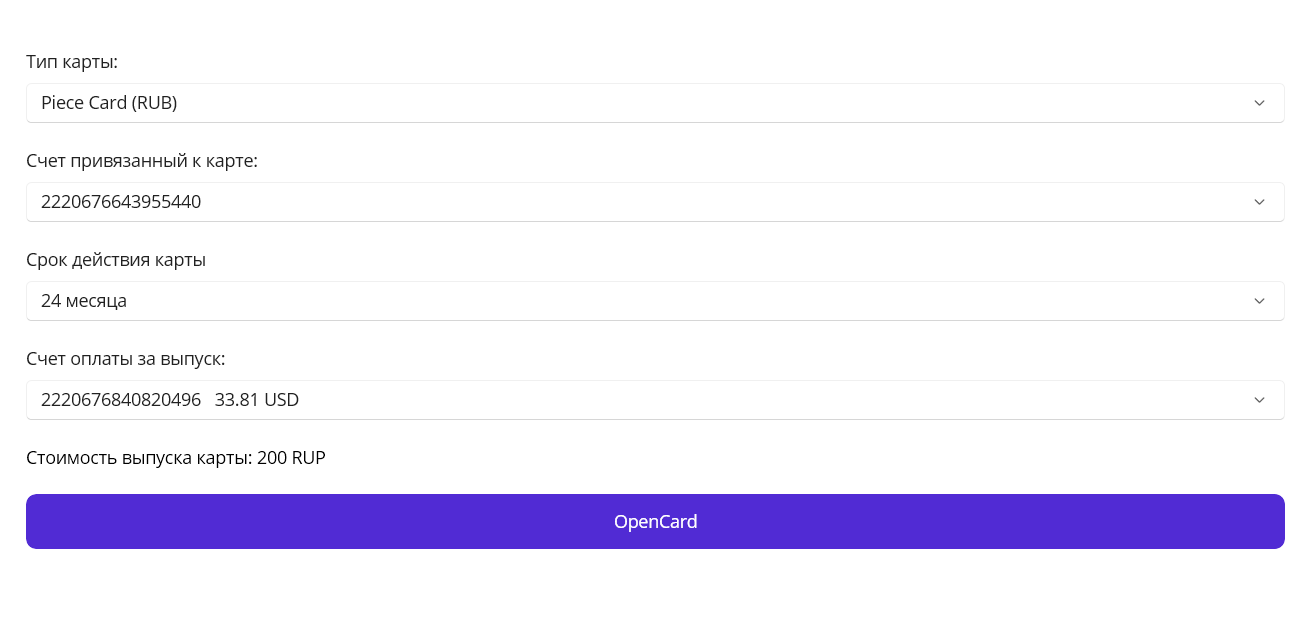


Рис. 9 Окно выпуска карты

Окно профиля

Одним из важнейших окон является окно профиля, представленное на рисунке 10. Здесь отображаются последние совершенные транзакции. А также функциональная кнопка выхода из приложения, после которой сбрасывается соединение с сервером и появляется возможность авторизации под другим аккаунтом.

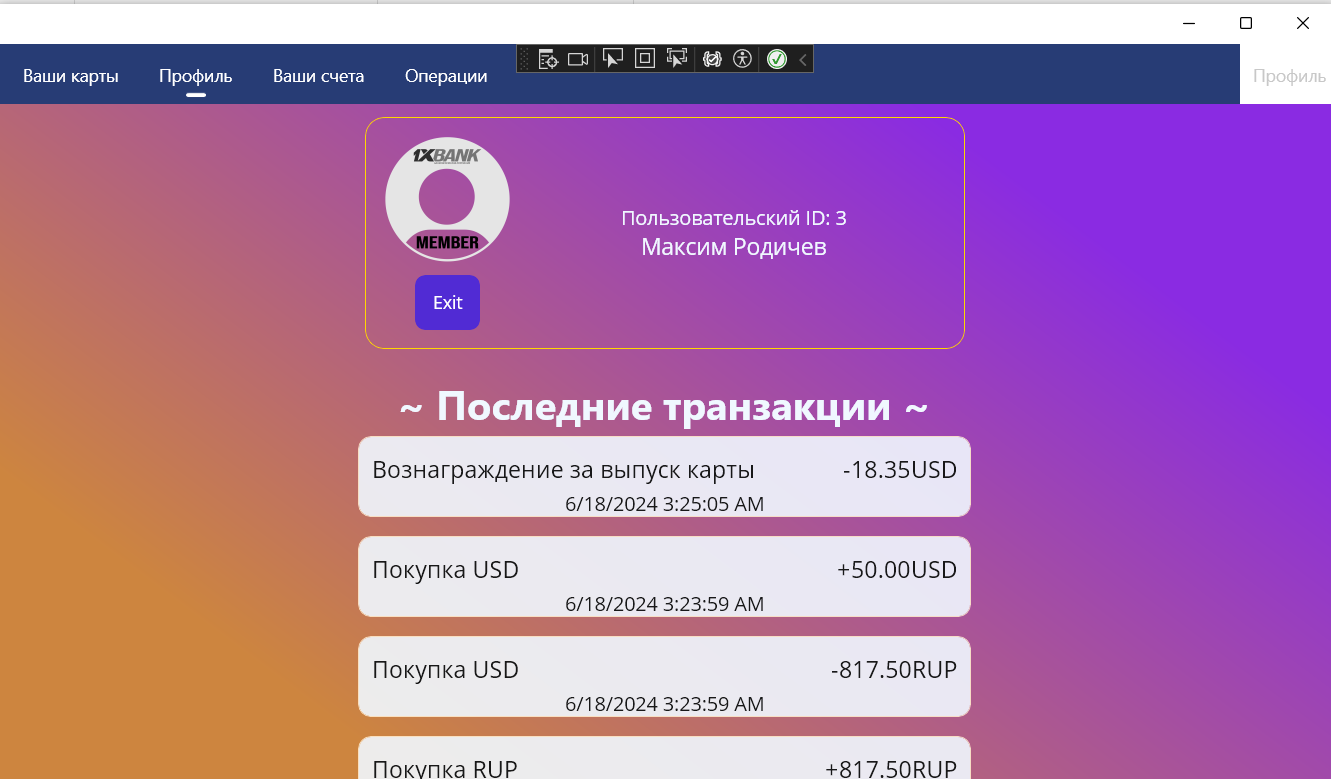


Рис. 10 Окно профиля

По нажатию на любую из транзакций можно получить краткую справку о данной транзакции, её id, счет отправителя, получателя и сумму транзакции, что показано на рисунке 11.

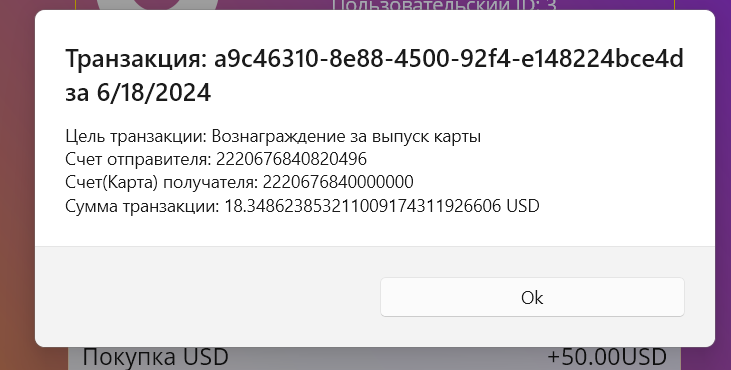


Рис. 11 Краткая справка о транзакции

Окно отображение счетов

Наряду с окно отображения карт, есть окно отображение счетов пользователя, что видно на рисунке 12.

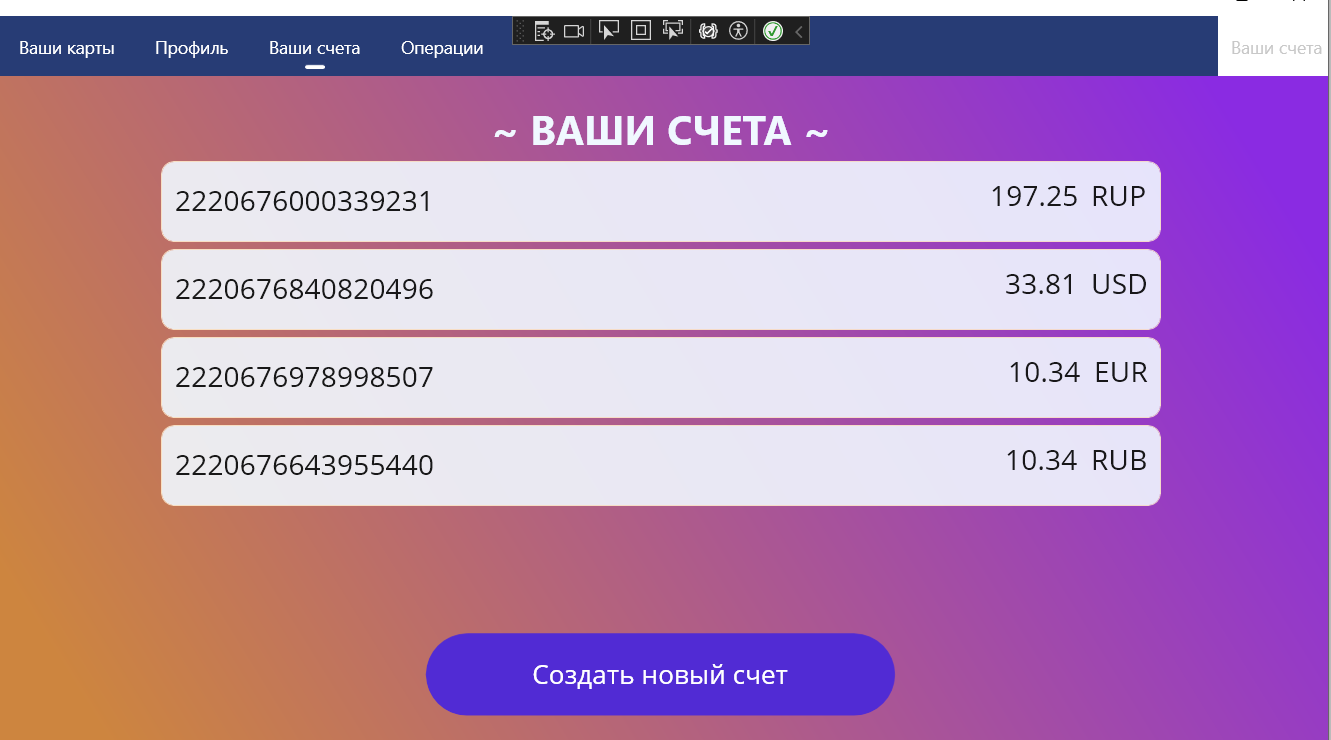


Рис. 12 Окно отображение счетов

По нажатию на любой из счетов, открывается окно взаимодействия с счетом (рисунок 13). Здесь показана справочная информация о данном счете и возможность получения транзакций, совершенных с данным счетом.

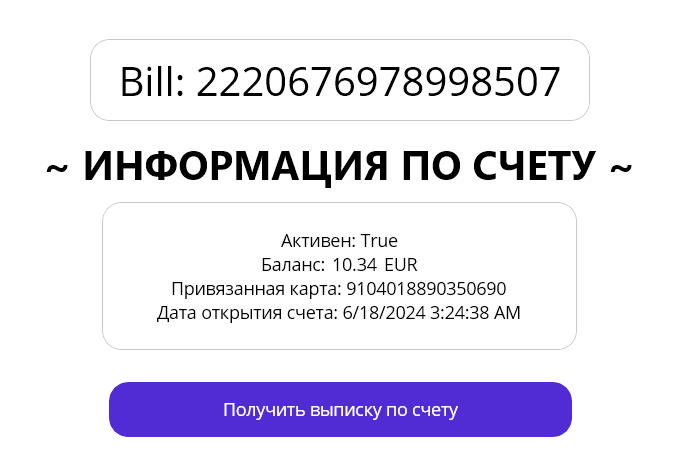


Рис. 13 Окно взаимодействия с счетом

По нажатию на кнопку «Создать новый счет» появляется окно диалога с пользователем. Здесь ему следует выбрать валюту с которой будет открыт счет. Смотреть рисунок 14.

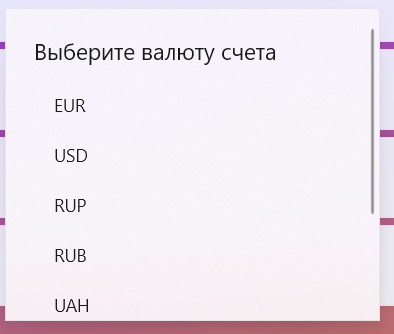


Рис. 14 Диалог с пользователем, выбор валюты счета

Окно операций

Последним окном остается функциональное окно, предоставляющее возможность отправки денежных средств и их конвертацию. Смотреть на рисунок 15.

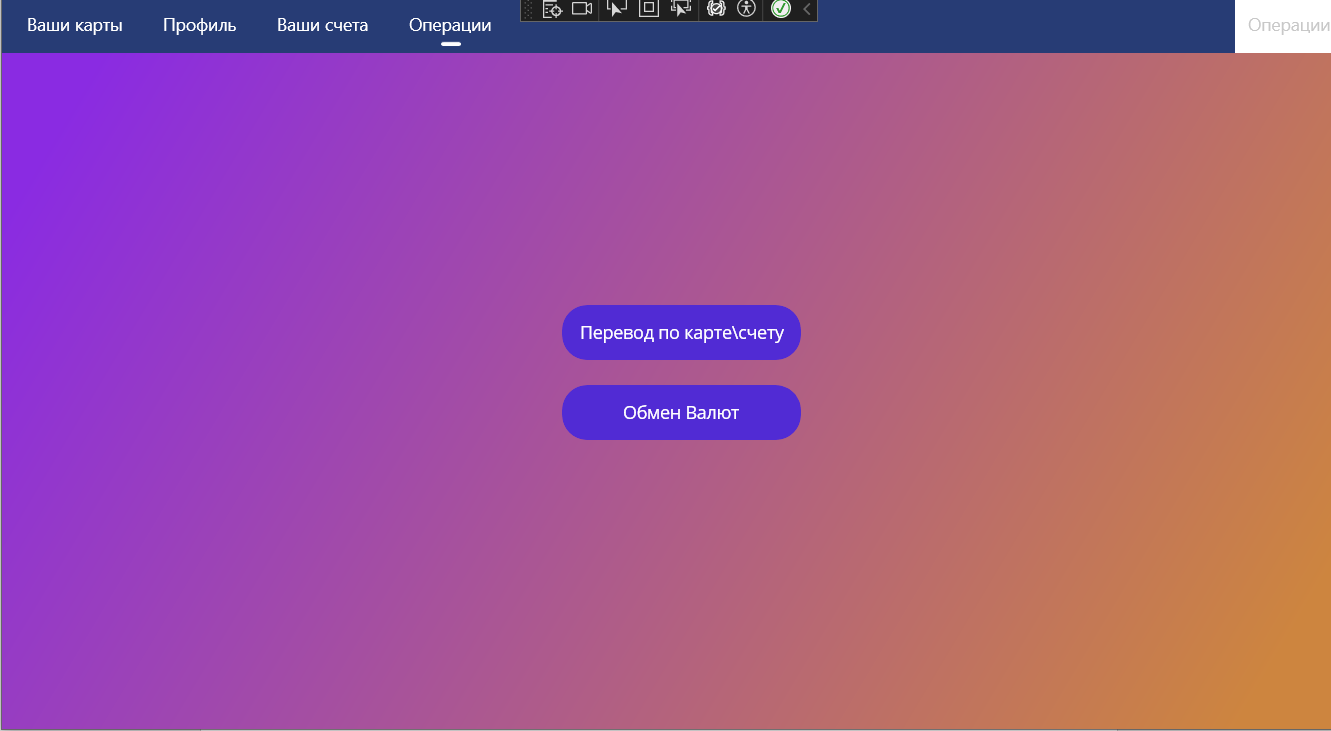


Рис. 15 Окно операций

«Перевод по карте\счету» вызывает диалог с пользователем, где требуется выбрать с какого счета будет совершаться перевод, ввести карту или счет пользователя, которому желаете отправить, в конце ввести сумму вашего перевода. Данный диалог продемонстрировать на рисунке 16.

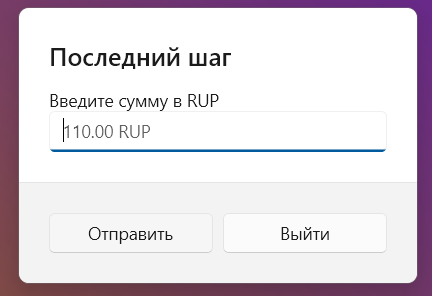
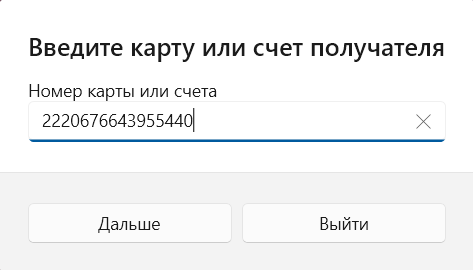
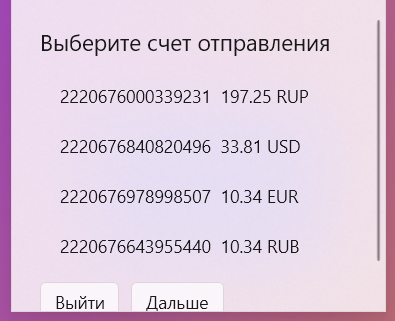
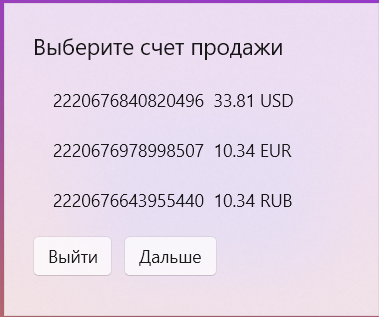
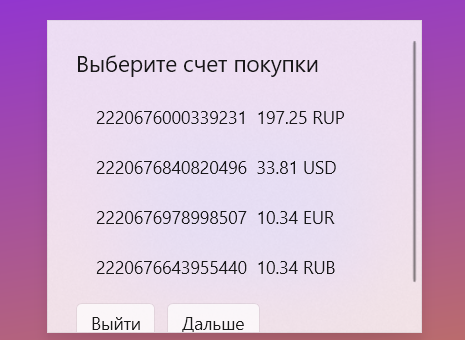


Рис. 16 Диалог «Перевод по карте\счету»

«Обмен валют» вызывает диалог с пользователем, где требуется выбрать из списка счет покупки и продажи, а так же ввести сумму покупки. Данный диалог продемонстрирован на рисунке 17.



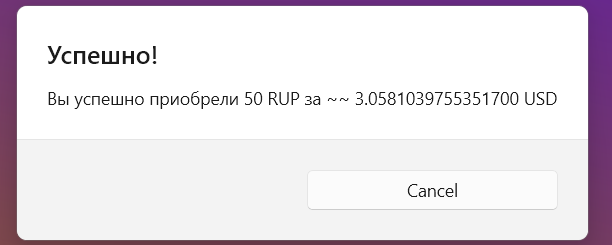
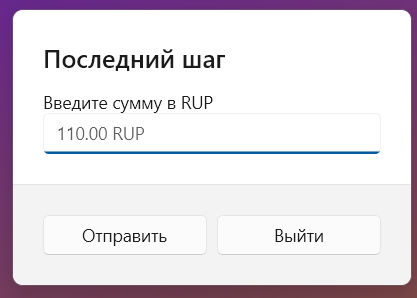


Рис. 17 Диалог «Обмен валют»

Это основное руководство пользователю, которое поможет ознакомиться с функциональностью и использованием приложения банковской системы.

## 2.4. Руководство программисту

Данная банковская система разработана для автоматизации деятельности банка и состоит из двух частей серверная и клиентская.

Минимальные технические требования к клиентскому приложению:

**Операционная система:** Windows 10;

**Процессор:**

* Минимум: AMD Ryzen 3 3200U или Intel Core i3-10110U;
* Рекомендуется: AMD Ryzen 3 5500U или Intel Core i5-10210U;

**Оперативная память:**

* Минимум: 8 ГБ;
* Рекомендуется: 16 ГБ;

**Место на диске:**

* Минимум: 512 МБ;
* Рекомендуется: 1 ГБ;

**Видеокарта:**

* Встроенная графика, совместимая с DirectX 11 или выше;

**Сетевое подключение:**

* Стабильное интернет-соединение для работы с серверной частью и онлайн-операциями;

Разработка клиентской и серверной части велась на следующей конфигурации:

Процессор: AMD Ryzen 3 5500U

Оперативная память: 16 ГБ DDR4 3200MHz

Хранилище: 256 ГБ SSD

Операционная система: Windows 10

Во время разработки были использованы следующие приложения:

– интегрированная среда разработки Microsoft Visual Studio 2022;

– объектно-ориентированные язык программирования C#;

– библиотеки: Microsoft.Maui.Controls, System, Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.Sqlite, Microsoft.AspNetCore.Authentication.JwtBearer, Microsoft.Maui.Controls, System.IdentityModel.Tokens.JWT;

– .NET MAUI, ASP.NET Core

– База данных: Sqlite (EntityFrameworkCore)

Для взаимодействия приложения с сервером важно учитывать его структуру.

Базы данных:

1. Таблица «Bills». Содержит информацию о счетах: номер счета, валюта счета, клиентский id, баланс счета, дата создания, дата закрытия (null), активность счета, привязанная карта к счету.

2. Таблица «Cards». Содержит информацию о картах: номер карты, тип карты, клиентский id, дата открытия, дата закрытия (срок действия), PIN, CVC коды, а так же главный счет данной карты.

3. Таблица: «Transactions». Содержит информацию о транзакциях: уникальный id транзакции, клиентский id отправителя и получателя, цель транзакции, сумма транзакции, дата создания транзакции.

4. Таблица: «Users». Содержит информацию о пользователях: уникальный клиентский id, роль пользователя, имя пользователя, фамилия пользователя, почта, пароль, дата создания аккаунта.

Взаимосвязи таблиц и их поля представлены на рисунке 18.

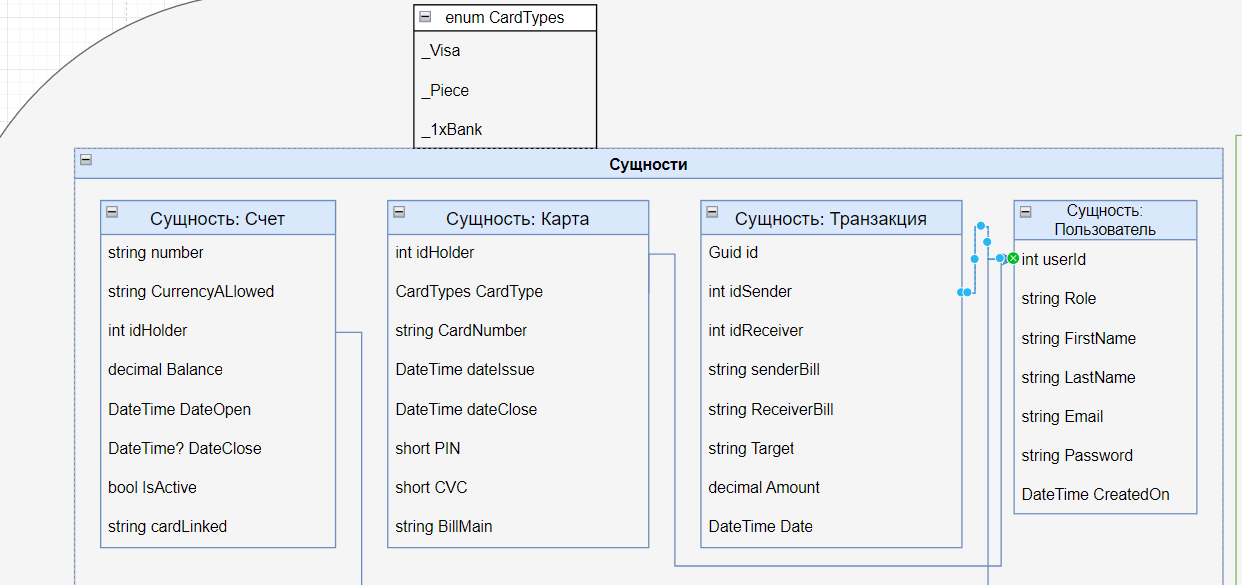


Рис. 18 Взаимосвязи сущностей, заносимых в БД

Для осуществления запросов на сервер используются http-запросы, список которых представлен ниже:

1. UsersController:

«{hostname}/users/registration». Запрос на регистрацию, принимает RegistrationDTO, возвращает ActionResult.

«{hostname}/users/login». Запрос на авторизацию и аутентификацию, принимает LoginDTO, возвращает JWT токен, позволяющий получить доступ к следующим контроллерам сервера.

2. CardController:

«{hostname}/card/open». Запрос на создание карты, принимает CardDTO, а так из header http запроса токен авторизации, по которому авторизует запрос и идентифицирует пользователя, желающего создать карту.

«{hostname}/card/get». Запрос на получения всех карт пользователя, отправляющего этот запрос.

«{hostname}/card/close». Запрос на закрытие карты, принимает CardDTO

«{hostname}/card/secureData/{number}». Запрос на получения секретных данных карты, принимает из тела запроса номер карты. Возвращает строковое представление PIN и CVC кодов.

3. BillController:

«{hostname}/bill/open/{currency}». Запрос на создание счета, принимает из тела запроса требуемую валюту.

«{hostname}/bill/get». Запрос на получение всех счетов данного пользователя.

4. KursController:

«{hostname}/kurs/get». Запрос на получения текущего курса валют.

5. TransactionController:

«{hostname}/transaction/get». Запрос на получения транзакций, связанных с данным пользователем.

6. PaymentController:

«{hostname}/payment/send». Запрос на отправку денежных средств, принимает PaymentDTO, в котором и прописываются все нужные инструкции по совершению транзакции.

Для более удобного представления о работы контроллерах серверной части, вниманию программиста предоставлен рисунок 19.

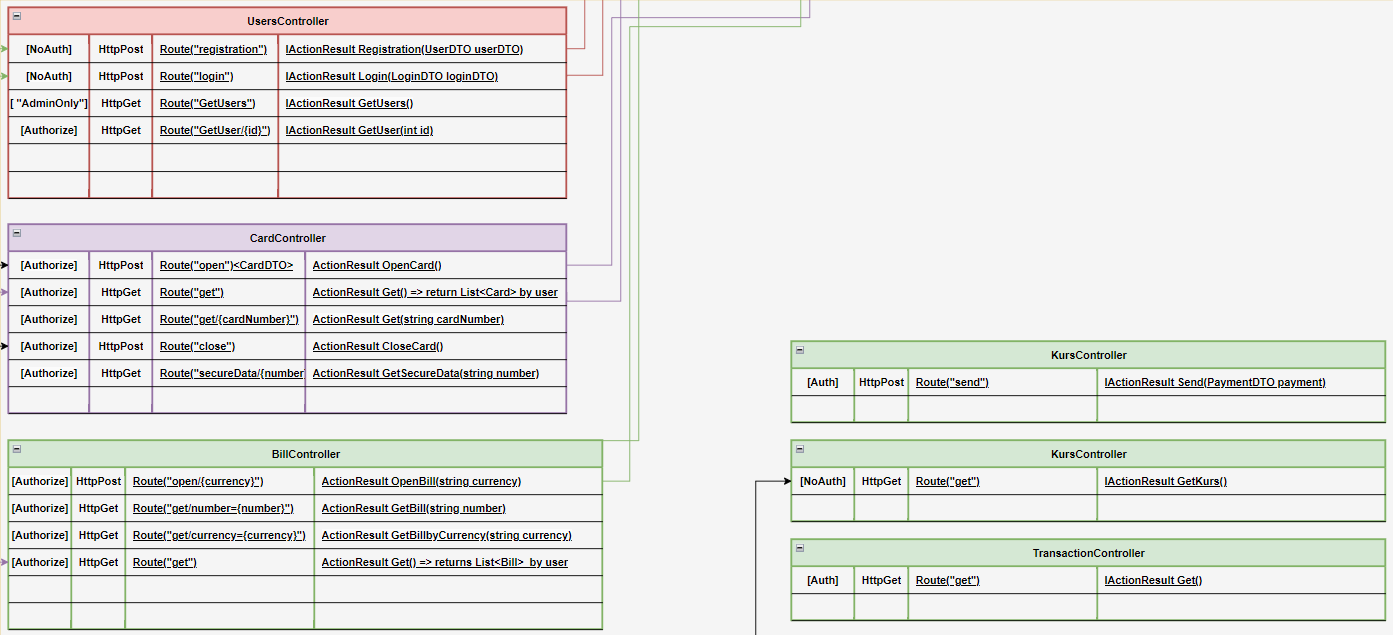


Рис. 19 Контроллеры сервера

Чтобы взаимодействовать с серверной часть, требуется понимать какие сущности он принимает или отдает, они представлены как Data-Transfer objects на рисунке 20.

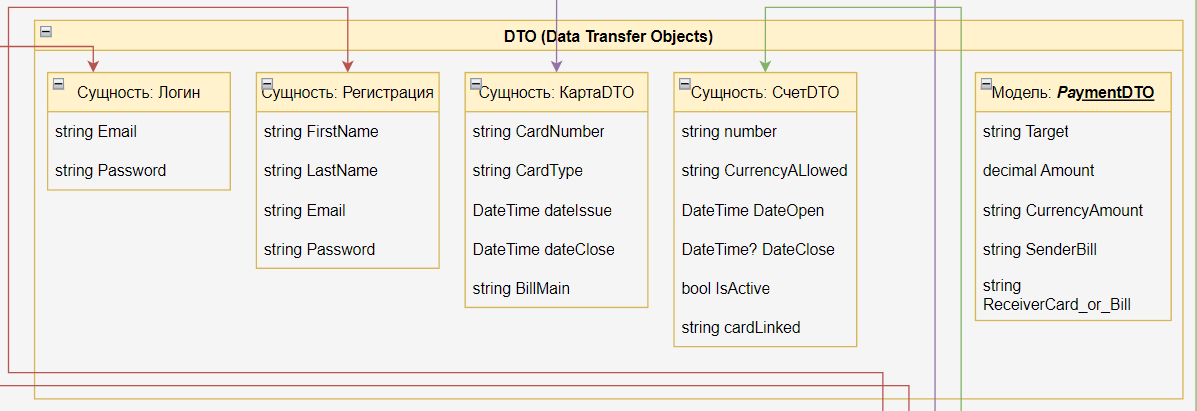


Рис. 20 Сводка DTO’s серверной части

Листинг кода из файлов с расширением .cs представлен в приложении.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана банковская система, включающая в себя клиентское приложение и серверную часть. Используя язык программирования C# и функционал .NET 8.0 (.NET MAUI & ASP.NET Core), а также базу данных Sqlite, удалось создать удобную навигацию в клиентской части и масштабируемую серверную часть.

Интерфейс позволяет легко взаимодействовать клиенту с счетами и картами, позволяя отслеживать финансовую активность, благодаря истории транзакции. База данных EntityFrameworkCore.Sqlite выбрана в качестве хранилища данных, благодаря своей простоте использования и возможности хранить сущности и получать записи без sql запросов. В базе данных хранятся данные о картах, счетах, транзакциях и клиентах.

Основные преимущества разработанной системы включают:

1. Удобный и интуитивно понятный пользовательский интерфейс, обеспечивающий простоту и эффективность работы с активами в банке.

2. Проработанная безопасность, хэширование паролей и JWT токены обеспечивают безопасность данных клиентов на должном уровне.

В ходе работы также выявлены некоторые возможные области для дальнейшего улучшения системы. Некоторые из них включают:

1. Расширение функционала: добавление больших возможностей по аудиту активов клиентов. Возможность просмотр диаграмм и иных справочных сведений по финансовой активности счетов и карт клиента.

2. Расширение функционала сервера: добавление базы данных для хранения JWT токенов, чтобы иметь возможность их обновлять и уменьшить срок действия.

Цель работы была успешно достигнута — разработана функциональная банковская система, демонстрирующая клиент-серверную архитектуру приложений.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Научная и учебно-методическая литература**

1. Лок Э. «ASP.Net Core In Action», Пер. Беликов Д. Москва: Manning, 2024. - 1050 с. - ISBN 978-5-93700-183-2.

2. Рихтер, Дж. «CLR via C#». Программирование на платформе Microsoft .NET Framework 4.5 на языке C#.

3. Прайс М. «C# 10 и . NET 6. Современная кросс-платформенная разработка», 2022 – 848с. - ISBN: 978-5-4461-2249-3.

**Интернет-ресурсы**

4. «Complete Guide to ASP.NET Core Identity»

(<https://frankliucs.com/courses/complete-guide-to-asp-net-core-identity/>)

5. «Master ASP.NET Core MVC (.NET 8) by building a Supermarket Management System» (<https://frankliucs.com/courses/master-net-core-mvc-by-building-a-supermarket-management-system-in-net-8/>)

6. «Справочник по C#» (<https://metanit.com/sharp/>)

7. «Форум программистов» (<https://stackoverflow.com/>)

8. «Пять простых шагов для понимания JSON Web Tokens (JWT)» (<https://habr.com/ru/articles/340146/>)