**Министерство профессионального образования и занятости**



**населения Приморского края**

**краевое государственное автономное**

**профессиональное образовательное учреждение**

**«ПРИМОРСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

**(КГА ПОУ «ППК»)**

Отделение Экономики и сервиса

Специальность *09.02.07*

«Информационные системы и программирование»

Работа допущена к защите:

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

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

**КУРСОВОЙ ПРОЕКТ**

**по теме:**

**РАЗРАБОТКА БАНКОВСКОГО ПРИЛОЖЕНИЯ**

|  |  |
| --- | --- |
| **Выполнил**  студент группы 1125 | **Руководитель**  преподаватель |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Куранаков | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Пушкин |

**Владивосток 2023**

**Министерство профессионального образования и занятости**



**населения Приморского края**

**краевое государственное автономное**

**профессиональное образовательное учреждение**

**«ПРИМОРСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»**

**(КГА ПОУ «ППК»)**

**Отделение Экономики и сервиса**

**З А Д А Н И Е**

на курсовой проект

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

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

|  |
| --- |
| *студенту 1125 группы* |

(фамилия, имя, отчество)

на тему «Разработка банковского приложения»

|  |
| --- |
|  |
|  |

Вопросы, подлежащие разработке (исследованию):

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

Основные источники информации и прочее, используемые для разработки темы

|  |
| --- |
| 1. Стиллмен Э., Грин Дж. – Изучаем C#. Включая C# 6.0 |
| 1. Мак–Дональд Мэтью – Windows Presentation Foundation в .NET 4.5 |
| 1. Виктор Гольцман – Библиотека программиста, MySQL 8.0 |
|  |

Срок предоставления работы «19» мая 2022 г.

Дата выдачи задания «01» апреля 20

2 г.

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

(подпись) (и.о. фамилия)

Задание получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.М. Куранаков

(подпись) (и.о. фамилия)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc103698377)

[1 Разработка системы работы приложения 5](#_Toc103698378)

[1.1 Требования к разрабатываемой системе 5](#_Toc103698379)

[1.2 Выявление основных сущностей и их взаимосвязей. 9](#_Toc103698380)

[1.3 Разработка диаграмм деятельности. 11](#_Toc103698381)

[1.4 Разработка схемы базы данных 13](#_Toc103698382)

[2 Описание разработанной системы 20](#_Toc103698383)

[2.1 Описание интерфейса пользователя 20](#_Toc103698384)

[2.1.1 Окно входа 20](#_Toc103698385)

[2.1.2 Окно регистрации 21](#_Toc103698386)

[2.1.3 Основное окно приложения 23](#_Toc103698387)

[2.1.3 Страница настроек 24](#_Toc103698388)

[2.1.4 Страница перевода валют 26](#_Toc103698389)

[2.1.5 Страница денежных переводов 27](#_Toc103698390)

[2.1.6 Страница просмотра расходов 29](#_Toc103698391)

[2.1.7 Окно предупреждения 30](#_Toc103698392)

[2.2 Пользовательские сценарии 30](#_Toc103698393)

[ЗАКЛЮЧЕНИЕ 32](#_Toc103698394)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 33](#_Toc103698395)

# ВВЕДЕНИЕ

Колледж — среднее специальное учебное заведение, реализующее основные профессиональные образовательные программы среднего профессионального образования базовой подготовки и программы среднего профессионального образования углублённой подготовки.

В эру развития информационных технологий появляется все больший спрос на обучение в колледже. Современные технологии и доступ к интернету привели к значительному увеличению информации и возможностей для самообразования. Однако, несмотря на это, обучение в колледже остается востребованным и значимым.

Объектом данной курсовой работы является профессиональный колледж «ДВФУ», а предметом – моделирование системы управления современным колледжем.

Целью данного курсового проекта является разработка симулятора управляющего данным учебным заведением. Из этого вытекают следующие задачи:

1. Выявить основные сущности и их взаимосвязи,
2. Сформировать требования к разрабатываемой системе,
3. Определить основные алгоритмы, описывающие работу учебного заведения,
4. Разработать приложение, которое моделирует поведение системы обучения,
5. Протестировать разработанное приложение.

# 1 Разработка системы работы приложения

## 1.1 Требования к разрабатываемой системе

Для начала необходимо разобраться, что такое функциональные требования.

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

Функциональные требования, как правило, состоят из:

* User story – показывает, чего вы ожидаете от команды разработки,
* Use cases – показывают сценарии использования,
* Wireframes – средство визуализации своей идеи.

Сосредоточимся на User story и Use cases.

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

Пользовательские истории помогают сосредоточиться на потребностях пользователя: как он будет использовать приложение? Чего он ждет от продукта? Как поведет себя в той или иной ситуации? Таким образом, ответы на эти вопросы помогут создателям продукта решать настоящие проблемы клиентов. Вот несколько главных задач, для которых необходимо использовать User Stories:

* Организовать работу. Когда проект разбит на части, связанные с пользовательскими историями, каждая из них представляет собой цельную и понятную задачу. Так разработчики могут фокусироваться на каждой из них и получать измеримый результат,
* Cохранить фокус на пользователе. Конечно, разработка включает себя десятки сложных задач, связанных с техническими, финансовыми и другими вопросами. Однако юзер стори – это постоянное напоминание команде о тех, для кого этот продукт создается, и направляют их работу в нужное русло,
* Сплотить команду. Несмотря на то, что у каждого есть свои задачи, каждый понимает конечную цель и видит себя частью целого. Только работая сообща, можно достичь нужного пользователю результата, и пользовательские истории дают четкое понимание этого аспекта работы,
* Найти свежие решения. Команда старается придумать самый приятный и интересный способ решить задачу пользователя. Часто это приводит к появлению новых интересных идей и их воплощению. Результат – полезный и уникальный продукт.

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

* Заголовок – краткое описание истории,
* Я как – роль,
* Хочу – функционал,
* Для того, чтобы – польза.

Необходимо создать приложение, которое позволит конечному пользователю создать аккаунт в приложении, совершать денежные переводы и оплату счетов. Так же приложение должно позволять изменять данные аккаунта пользователя, вплоть до полного удаления. Помимо этого, в приложении должна быть функция перевода денег из одной валюты в другую.

На основе данных требований можно составить следующие user stories для разрабатываемого приложения:

* Пользователь хочет иметь возможность создать аккаунт в приложении, чтобы управлять своими финансами,
* Пользователь хочет иметь возможность следить за тем, как и куда он тратит свои деньги, чтобы лучше контролировать свои расходы,
* Пользователь хочет иметь возможность производить оплату за какие–либо услуги,
* Пользователь хочет иметь возможность совершать денежные переводы другим пользователям.

Use case (также юзкейс, сценарий использования) – это сценарий взаимодействия пользователя или пользователей с программным продуктом для достижения конкретной цели.

Юзкейсы содержат следующие сведения:

* кто использует сайт или приложение,
* что пользователь хочет сделать,
* цель пользователя,
* шаги, которые делает пользователь, чтобы совершить определенное действие,
* описание того, как сайт или приложение реагируют на действия пользователя.

Юзкейсы не содержат детали реализации, а также описания пользовательского интерфейса или экранов.

В общем, в юзкейсе описывается не каким образом программа делает что–либо, а что именно она делает. Именно этого подхода и нужно придерживаться, создавая юзкейсы.

В отличие от user story, которая излагается от имени какого–то конкретного пользователя, в use case может быть описано взаимодействие (с определенной целью) нескольких участников. Например:

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

Юзкейсы могут содержать следующие элементы. Их количество зависит от сценария:

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

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

Например, работа с системами типа «тасктрекер» задается достаточно простыми и стандартными Use Case: «Создать задачу», «Назначить задачу», «Пометить задачу, как выполненную». Однако тасктрекеров существует огромное множество, и это оправдано тем, что в каждом есть свои возможности по заданию жизненных циклов задач, их типов и взаимосвязей. И эту внутреннюю логику работы с задачами нет смысла описывать в виде Use Case.

## 1.2 Выявление основных сущностей и их взаимосвязей.

Даже хороших имен, указывающих пользователю, какую информацию стоит ожидать от сущности, обычно недостаточно. Каждая сущность нуждается в ясном, точном и полном описании или определении, чтобы быть однозначно интерпретируемой в рамках корпорации. Описание сущности должно объяснять смысл сущности и ее значение для корпорации.

Хотя описание, определение и назначение часто используются в качестве синонимов, термин описание предпочтительнее, поскольку он побуждает нас описывать сущности в терминах, понятных для пользователя.

Описание сущности должно объяснять ее смысл, а не то, как будет использоваться информация этой сущности. Необходимо собирать описания сущностей во время идентификации сущностей. При включении информации об использовании необходимо быть осторожным: подобная информация должна использоваться только в качестве примера или для пояснения. Способ использования информации изменяется более часто, чем информация сама по себе, поэтому информация об использовании непостоянна.

Описание сущности должно быть ясным, точным, полным и непротиворечивым. Оно должно быть сформулировано без привлечения технических терминов, понятно любому, кто хотя бы чуть–чуть знаком с описываемой концепцией.

Сущность – это множество экземпляров реальных или абстрактных объектов (человек, место, вещь, событие, состояние, концепция, идея, предмет и т.п.), обладающих общими атрибутами или характеристиками, и о которых необходимо хранить информацию. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникальной. При этом имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр. Сущности именуются обычно существительными, такими как «покупатель», «компьютер», «служащий», «продажа».

Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Диаграммы вариантов использования описывают функциональное назначение системы или то, что система должна делать. Разработка диаграммы преследует следующие цели:

* определить общие границы и контекст моделируемой предметной области,
* сформулировать общие требования к функциональному поведению проектируемой системы,
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей,
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Для того, чтобы более точно понять, как должна работать система, все чаще используется описание функциональности системы через варианты использования. Варианты использования – это описание последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей или других программных систем. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя, поэтому они точнее позволяют ранжировать функции по значимости получаемого результата.

Варианты использования предназначены в первую очередь для определения функциональных требований к системе и управляют всем процессом разработки. Все основные виды деятельности такие как анализ, проектирование, тестирование выполняются на основе вариантов использования. Во время анализа и проектирования варианты использования позволяют понять, как результаты, которые хочет получить пользователь влияют на архитектуру системы и как должны себя вести компоненты системы, для того чтобы реализовать нужную для пользователя функциональность.

Стратегия использования прецедентов при определении требований определяет необходимость дополнительно к вопросу "что пользователи ждут от системы?" задавать вопрос "что система должна сделать для конкретного пользователя?". Такой подход позволяет искать функции, которые нужны многим пользователям и исключать те возможности, которые не могут помочь пользователям выполнять свои повседневные задачи.

Определим сущность для разрабатываемого приложения. В данном случае в качестве актера выступает Пользователь, который имеет варианты взаимодействия с приложением.



Рисунок 1 – Use Case диаграмма приложения

## 1.3 Разработка диаграмм деятельности.

Диаграмма активностей (видов деятельности) – один из доступных видов диаграмм. Она отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок–схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

То есть если варианты использования ставят перед системой цель, то диаграмма деятельности показывает последовательность действий, необходимых для ее достижения. Действия (action) – это элементарные шаги, которые не предполагают дальнейшую декомпозицию.

Деятельность – это особая операция системы. Диаграммы действий используются не только для визуализации динамической природы системы, но они также используются для построения исполняемой системы с использованием методов прямого и обратного проектирования. Единственная недостающая вещь на диаграмме активности – это часть сообщения.

Он не показывает поток сообщений от одного действия к другому. Диаграмма деятельности иногда рассматривается как блок–схема. Хотя диаграммы выглядят как блок–схема, это не так. Он показывает разные потоки, такие как параллельный, разветвленный и одиночный.

Деятельность может содержать входящие или исходящие дуги деятельности, показывающие потоки управления и потоки данных. Если поток соединяет две деятельности, он является потоком управления. Если поток заканчивается объектом, он является потоком данных.

Деятельность выполняется, только тогда, когда готовы все его «входы», после выполнения, деятельность передает управление или данные на свои «выходы». Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали слева направо или сверху вниз.

В точку ветвления входит ровно один переход, а выходит – два или более. Для каждого исходящего перехода задается булевое выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение "истина", иначе поток управления окажется неоднозначным. Желательно чтобы условия покрывали все возможные варианты, иначе поток остановится.

Из этого мы имеем следующую диаграмму, описывающую алгоритм авторизации в приложении (Рисунок 2).

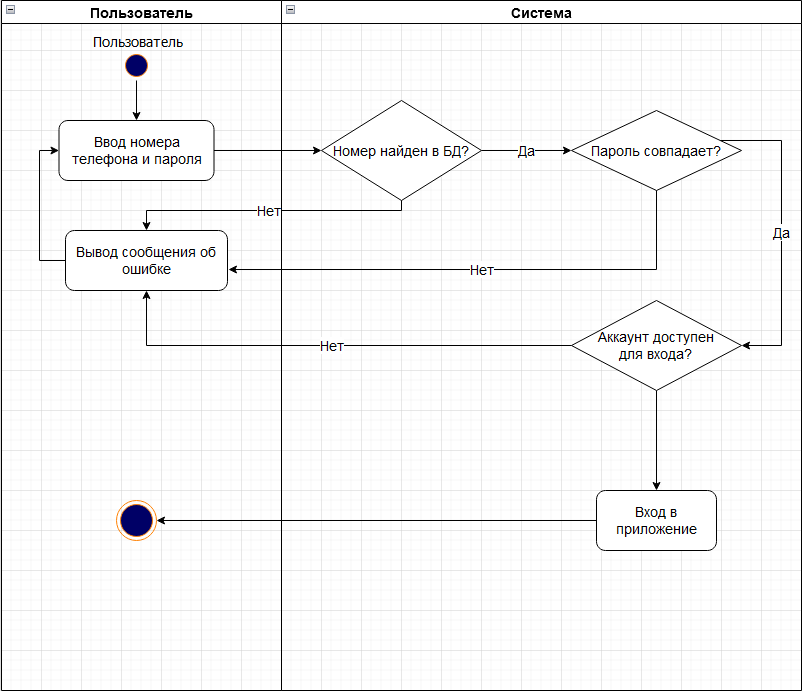


Рисунок 2 – Диаграмма активностей

## 1.4 Разработка схемы базы данных

База данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных. Данные вместе с системой управления базами данных, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов, формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов.

Обширный сбор данных из Интернета вещей меняет действительность и производственный сектор по всему миру: современные компании имеют доступ к большему количеству данных, чем когда–либо прежде. Прогрессивные компании теперь могут использовать базы данных, чтобы от обычного хранения данных и базовых транзакций перейти к анализу огромных объемов данных из множества систем. Благодаря базам данных и другим средствам вычислений и бизнес–аналитики современные компании могут использовать собираемые ими данные для более эффективной работы, эффективного принятия решений, гибкости и масштабируемости. Оптимизация доступа к данным и пропускной способности сегодня критически важна для бизнеса, потому что объем данных, которые нужно отслеживать, постоянно растет. Крайне важно иметь платформу, которая может обеспечить производительность, масштабируемость и гибкость, необходимые предприятиям по мере их роста с течением времени.

Схема базы данных представляет собой логическую конфигурацию либо целой реляционной базы данных, либо ее части. Схема может существовать как в виде наглядного представления базы данных, так и в виде набора формул (также именуемых «условиями целостности»), которые регулируют ее устройство. Эти формулы выражаются с помощью языка описания данных, например, SQL. Будучи частью словаря данных, схема показывает, как связаны между собой сущности, из которых состоит база данных: таблицы, представления, хранимые процедуры и так далее.

Схема обычно создается проектировщиком базы данных, чтобы помочь программистам, чье приложение будет взаимодействовать с данной базой. Процесс создания схемы базы данных называется моделированием данных. Если вы пользуетесь трехсхемным подходом к проектированию базы данных, этот шаг будет следовать за созданием концептуальной схемы. Стоит отметить, что в центре внимания концептуальной схемы находится не структура базы данных, а информационные нужды организации.

При проектировании базы данных стоит учитывать, что любая БД состоит из трёх моделей:

* Инфологическая модель,
* Даталогическая модель,
* Физическая модель.

Проектирование инфологической модели является основной задачей при создании БД. Цель инфологической модели – обеспечение наиболее естественных для человека способов сбора и представления той или иной информации, которую предполагается хранить в создаваемой базе. Поэтому инфологическую модель данных пытаются строить по аналогии с естественным языком. Последний не может быть использован в чистом виде из–за сложности компьютерной обработки текстов и неоднозначности любого естественного языка. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства.

Поэтому необходимо четко поставить цель данной БД, а также установить, какую информацию получает пользователь в результате работы с программой.

Проектирование даталогической модели – важный этап в проектировании БД. На этом этапе важно правильно выделить сущности и описать их атрибуты. Ошибка на этом этапе может обернуться разработчику значительными потерями времени и сил в дальнейшем.

Датологическая модель – модель, описывающая логику организации данных. Датологическое проектирование заключается в проектировании логической структуры БД. Таким образом, главное отличие даталогической модели от инфологической состоит в том, что инфологическая модель хранит в себе всю информацию о предметной области, необходимую и достаточную для проектирования базы данных, но она не привязана к определенной СУБД.

Физическая модель БД определяет способ размещения данных на носителях, устройствах внешней памяти, а также способ и средства организации эффективного доступа к ним. Поскольку СУБД функционирует в составе и под управлением операционной системы, то организация хранения данных и доступа к ним зависит от принципов и методов управления данными операционной системы.

В отличие от ранних СУБД, многие современные системы не предоставляют разработчику какого–либо выбора на этой стадии. Реально к вопросам проектирования физической модели можно отнести:

* выбор схемы размещения данных,
* определение числа и типа индексов.

Схему построенной базы данных можно увидеть на рисунке 3:

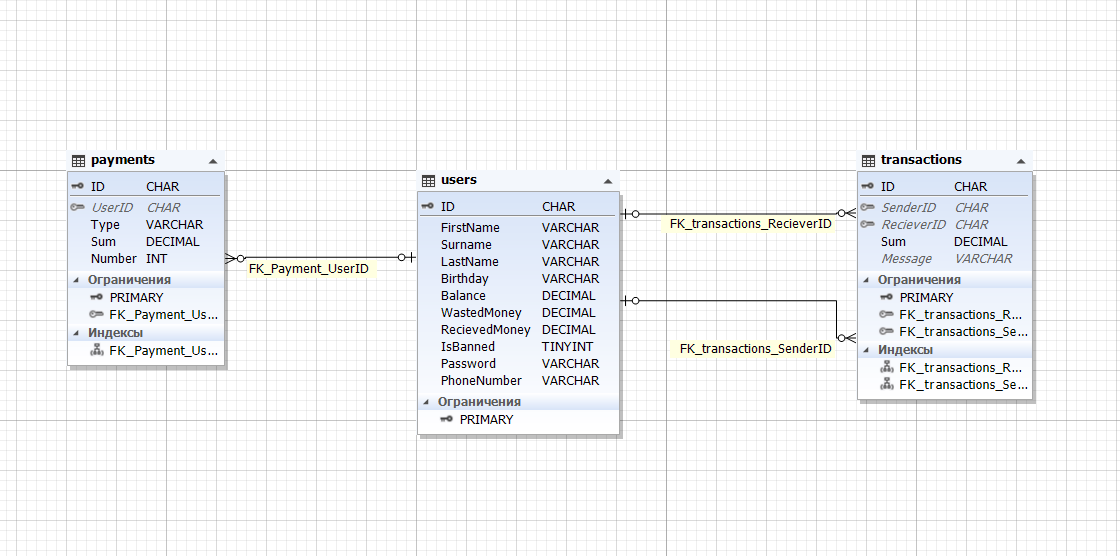


Рисунок 3 – Схема базы данных

В таблице users находятся данные о пользователях, зарегистрированных в приложении. Там хранится подробная информация о них, такая как их Ф.И.О., дата рождения, номер телефона, а также пароль, по которому пользователь входит в свой аккаунт. Помимо этого, туда помещены данные о финансах пользователя:

* Сколько денег на данных момент находится на счету,
* Сколько всего денег было получено пользователем за всё время использования приложения,
* Сколько денег в общей сумме было потрачено.

Отдельная строка в данной таблице выделена под данные о доступе к аккаунту пользователя – заблокирован он, или нет.

В таблице transactions хранятся данные о переводах, которые были совершены пользователями приложения, а именно:

* ID отправителя перевода,
* ID получателя,
* Сумма перевода,
* Сообщение, отправленное вместе с переводом.

В таблице payments хранятся данные об услугах, на которые пользователи тратили свои деньги. Там находится информация о типе услуги и сумме потраченных на данный тип услуг денег.

**1.5 Используемые технологии**

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. Сборка мусора автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда–выражения поддерживают приемы функционального программирования. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем. В C# имеется Единая система типов. Все типы C#, включая типы–примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

Вот лишь несколько функций языка C#, обеспечивающих надежность и устойчивость приложений: сборка мусора автоматически освобождает память, занятую уничтоженными и неиспользуемыми объектами. Обработка исключений предоставляет структурированный и расширяемый способ выявлять и обрабатывать ошибки, строгая типизация языка не позволяет обращаться к неинициализированным переменным, выходить за пределы индексируемых массивов или выполнять неконтролируемое приведение типов.

Чтобы обеспечить совместимость программ и библиотек C# при дальнейшем развитии, при разработке C# много внимания было уделено управлению версиями. Многие языки программирования обходят вниманием этот вопрос, и в результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Помимо кода на C# для разработки приложения используются базы данных, написанные на языке программирования SQL. База данных в SQL Server состоит из коллекции таблиц, в которой хранится определенный набор структурированных данных. Таблица содержит коллекцию строк, также называемых записями или кортежами, и столбцов, также называемых атрибутами. Каждый столбец в таблице предназначен для хранения конкретного типа данных, например дат, имен, денежных сумм или чисел.

MySQL, это система управления реляционными базами данных. В реляционной базе данных данные хранятся не все вместе, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов, а также наиболее распространенный стандартный язык, используемый для доступа к базам данных.

Большинство программистов отдают предпочтение MySQL, который является очень быстрым, надежным и легким в использовании. MySQL обладает также рядом удобных возможностей, разработанных в тесном контакте с пользователями. Первоначально сервер MySQL разрабатывался для управления большими базами данных с целью обеспечить более высокую скорость работы по сравнению с существующими на тот момент аналогами. И вот уже в течение нескольких лет данный сервер успешно используется в условиях промышленной эксплуатации с высокими требованиями. Несмотря на то, что MySQL постоянно совершенствуется, он уже сегодня обеспечивает широкий спектр полезных функций. Благодаря своей доступности, скорости и безопасности MySQL очень хорошо подходит для доступа к базам данных по сети интернет.

Так же стоит упомянуть о Nuget–пакетах, или же библиотеках, которые были использованы в процессе разработки приложения:

* Font Awesome WPF: библиотека для разработки графических приложений, содержащая набор различных иконок, необходимых для создания качественного и удобного пользовательского интерфейса,
* My Sql Data: библиотека классов, необходимых для разработки системы получения и записи данных в базу данных MySQL с помощью кода на языке программирования C#,
* Property Changed Fody: библиотека, упрощающая написание программного кода графического приложения с использованием паттерна разработки MVVM.

# 2 Описание разработанной системы

## 2.1 Описание интерфейса пользователя

Пользовательский интерфейс (user interface или сокращенно UI) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. UI должны быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне. Интерфейсы программного обеспечения также называют графическими пользовательскими интерфейсами. Пользовательские интерфейсы бывают жестовые, тактильные, голосовые, графические, командной строки и даже нейронные.

Ниже приведено описание интерфейса разработанного приложения

### 2.1.1 Окно входа

При входе в приложение пользователя встречает окно входа, видимое на рисунке 4.

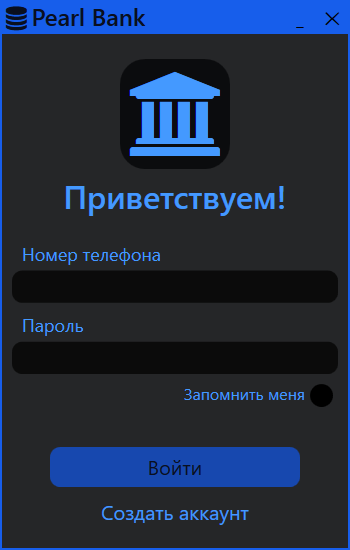


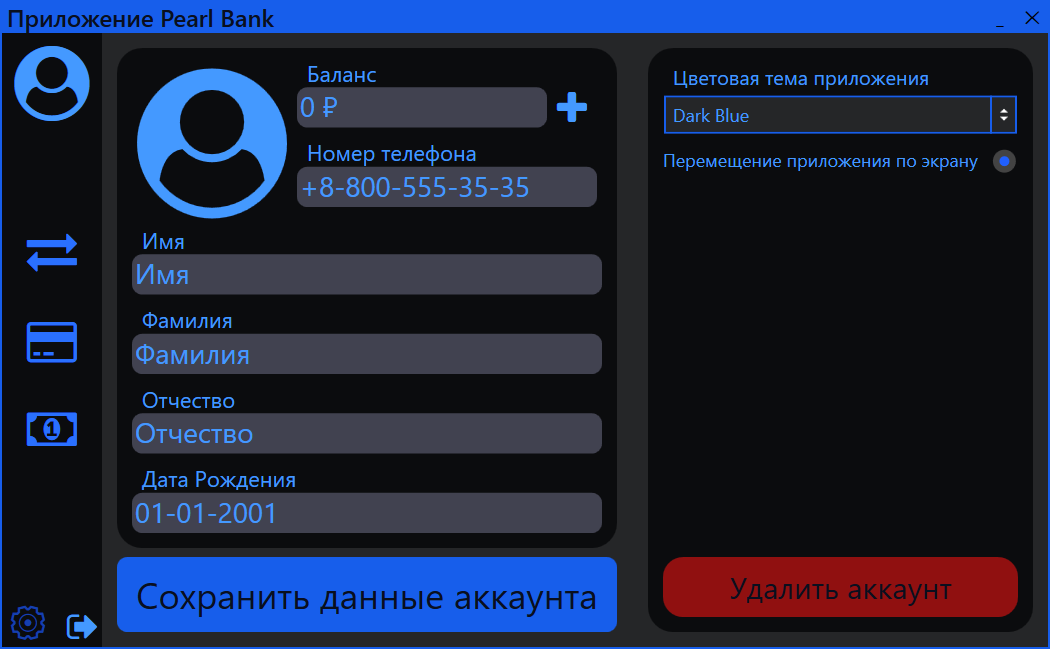
Рисунок 4 – Окно входа

Окно содержит эмблему банка, расположенную в верхней части окна, ровно по центру, приветственный текст, два текстовых поля, подписанные как «Номер телефона» и «Пароль», необходимые для осуществления входа в приложение. Чуть ниже, в правой части экрана, находиться небольшой флажок с надписью «Запомнить меня» – если этот флажок находится в активном состоянии, программа запомнит введённые пользователем данные, и при следующем запуске они автоматически будут введены в текстовые поля. В противном случае они будут оставаться пустыми.

В верху окна находится небольшая панель, на которой располагается название банка, и две кнопки, которые позволяют свернуть приложение, либо же выйти из него.

В нижней части окна, друг над другом, находятся две кнопки – «Войти» и «Создать аккаунт». Кнопка с надписью «Войти» не активна и подсвечена бледным оттенком синего цвета, до тех пор, пока в поля с именами «Номер телефона» и «Пароль» не будут введены какие–либо символы – в таком случае она будет выделена более ярко. После нажатия кнопки «Войти», если по введённому пользователем номеру телефона в базе данных найден аккаунт, и введённый пароль совпадает с найденным, будет осуществлён вход в приложение. Иначе на экран будет выведено сообщение об ошибке, с предупреждением о том, что такого номера телефона не существует в базе данных, либо введённый пароль не совпадает с паролем, установленным для найденного аккаунта.

На случай, если пользователь ещё не имеет аккаунта в приложении и желает его создать, ниже находится кнопка с надписью «Создать аккаунт», при нажатии на которую на экран будет выведено окно регистрации.



### 2.1.2 Окно регистрации

Это окно, расположенное на рисунке 5, визуально разделено на две разные части – в левой части находятся эмблема банка, текст приветствия для пользователя и, в самом низу, кнопка с надписью «Уже есть аккаунт?», которая, при нажатии, вновь выведет на экран окно входа в приложение. Эта функция нужна на случай, если пользователь случайно нажал на кнопку регистрации, либо вспомнил, что у него уже есть аккаунт, и он хочет в него войти.

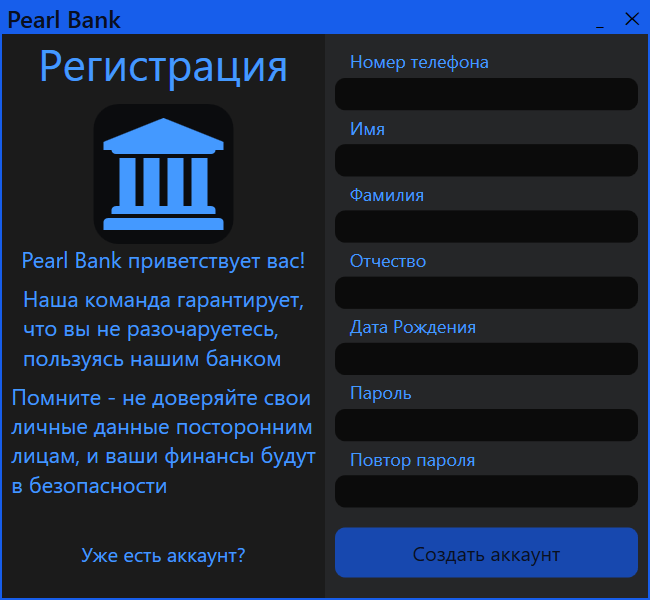


Рисунок 5 – Окно регистрации

В правой же части окна находится семь текстовых полей, для ввода информации о пользователе, такой как:

* Номер телефона,
* Имя,
* Фамилия,
* Отчество,
* Дата рождения,
* Пароль,
* Повтор пароля.

Поля «Повтор пароля» необходимо для того, чтобы убедиться, что пользователь точно помнит пароль, который хочет установить для своего аккаунта, и сможет войти в него позже.

Вверху окна вновь расположена панель с названием банка, и двумя кнопками «Свернуть» и «Закрыть».

Внизу расположена кнопка «Создать аккаунт», которая будет неактивной до тех пор, пока не будут заполнены все текстовые поля, и при её нажатии, при условии, что введённые пользователем данные подходят по формату, будет создан новый аккаунт и осуществлён вход.

### 2.1.3 Основное окно приложения

После входа в приложение, будет вызвано основное окно, в котором будет активна страница настроек (Рисунок 6).

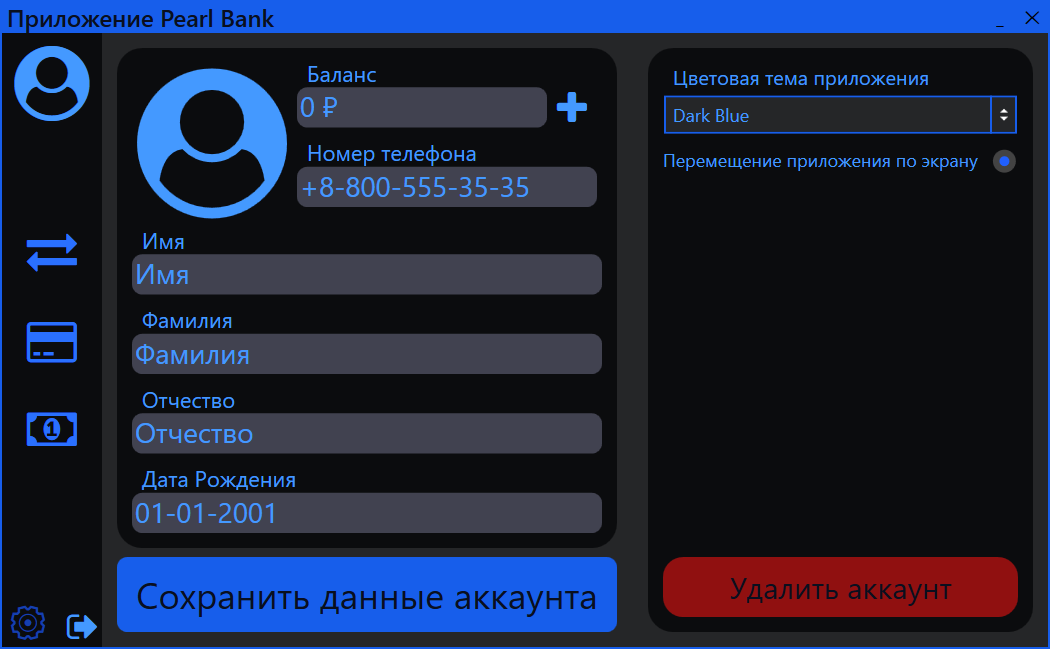


Рисунок 6 – Главное окно и страница настроек

В самом верху окна расположена панель синего цвета, в левой части которой находится надпись «Приложение Pearl Bank», а в правой – две кнопки, которые позволяют свернуть приложение, либо закрыть его.

В левой части находится панель навигации по приложению – она более тёмного цвета, чем основной фон и на ней расположено несколько различных кнопок, необходимых для навигации по страницам приложения. В самом центре панели расположены три кнопки:

* Кнопка с двумя стрелочками, направленными в противоположные стороны. Нажатие на эту кнопку ведёт на страницу, на которой можно осуществить перевод на имя другого пользователя, либо оплатить какую–либо услугу по номеру счёта,
* Кнопка с иконкой банковской карты ведёт на страницу просмотра расходов. Там можно увидеть, сколько всего денег было потрачено либо получено пользователем, список полученных и отправленных пользователем переводов, и все его траты, разбитые на 10 категорий,
* Кнопка с иконкой денежной купюры ведёт на страницу обмена валют, где можно по действующему курсу перевести деньги в рублях в доллары или евро, и наоборот.

Внизу этой панели находятся ещё две функциональные кнопки:

* Кнопка с иконкой шестерёнки, находящаяся в левом углу, ведёт на страницу настроек, на которой можно изменить данные аккаунта, либо настройки самого приложения,
* Кнопка «Выйти из аккаунта», находящаяся правее, при нажатии закрывает приложение и выводит на экран окно входа.

При наведении курсора на любую из вышеперечисленных кнопок, они будет подсвечены другим цветом, что обеспечивает более приятный визуальный отклик от приложения. Так же кнопка, которая соответствует странице, на которой находится пользователь, всегда выделена более тёмным оттенком – это необходимо для более простой навигации по страницам приложения.

### 2.1.3 Страница настроек

На странице настроек, которую можно увидеть на рисунке 6, содержится информация о данных аккаунта пользователя и текущих настройках приложения.

В правой части страницы находится форма, отвечающая за настройки приложения. В верху этой формы находится выпадающий список, с помощью которого можно изменить действующую цветовую тему приложения на одну из доступных на данный момент:

* Тёмно–синяя,
* Тёмно–фиолетовая.

Ниже списка находится флажок, отвечающий за переключение такой возможности, как перемещение окна приложения по экрану с помощью нажатия левой кнопкой мыши по верхней панели окна.

В нижней части формы находится красная кнопка, позволяющая пользователю навсегда удалить свой аккаунт из приложения на случай, если он больше не нуждается в услугах банка. При наведении она подсвечивается ярко–красным, предупреждая о возможных последствиях нажатия данной кнопки.

В левой части страницы находится большая форма, в которой хранятся такие данные о пользователе как:

* Баланс на текущий момент,
* Номер телефона,
* Имя,
* Фамилия,
* Отчество,
* Дата рождения.

Все эти данные находятся в полях для ввода, с помощью которых их можно изменять в любой момент времени по желанию пользователя. Кроме поля «Баланс» – оно недоступно для свободного ввода туда какого–либо числа, вместо этого рядом с полем находится кнопка «+», благодаря которой и можно пополнить счёт прямо внутри приложения. Внизу, под этой формой, находится большая кнопка «Сохранить данные аккаунта». При нажатии на эту кнопку информация, изменённая пользователем, будет сохранена и корректно обновлена в базе данных. Однако, если одно из полей окажется пустым, кнопка не будет активна, пока оно не будет заполнено

### 2.1.4 Страница перевода валют

При нажатии на кнопку с иконкой купюры, пользователю будет показана страница перевода валют, показанная на рисунке 7.

На данной странице находятся четыре формы, с помощью которых можно перевести некоторое количество одной доступной валюты в другую. В каждой форме, ровно по центру, расположена эмблема банка, ниже неё расположено два текстовых поля, по левому и правому краю формы. Над каждым из полей находится значок, который соответствует валюте, значение которой в нём расположено. В левое поле вводится значение валюты, которую пользователю необходимо перевести, значение в правом поле недоступно для ручного ввода, вместо этого оно рассчитывается автоматически, исходя из введённого пользователем количество первой валюты, и курсу, по которому осуществляется перевод. В верхней части формы находятся два значка, обозначающие валюты, которые можно перевести с помощью этой формы, и значок перевода – две стрелочки, направленные противоположно друг другу.

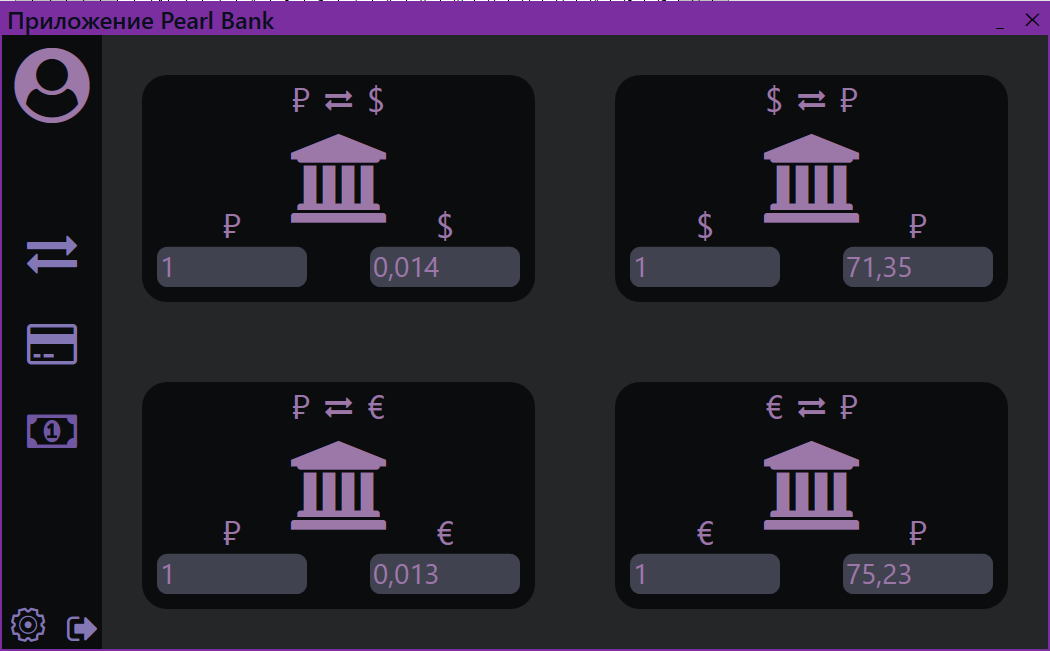


Рисунок 7 – Страница перевода валют

### 2.1.5 Страница денежных переводов

После нажатия на кнопку с двумя стрелочками, расположенную на панели навигации, пользователь окажется на странице денежных переводов, показанной на рисунке 8.

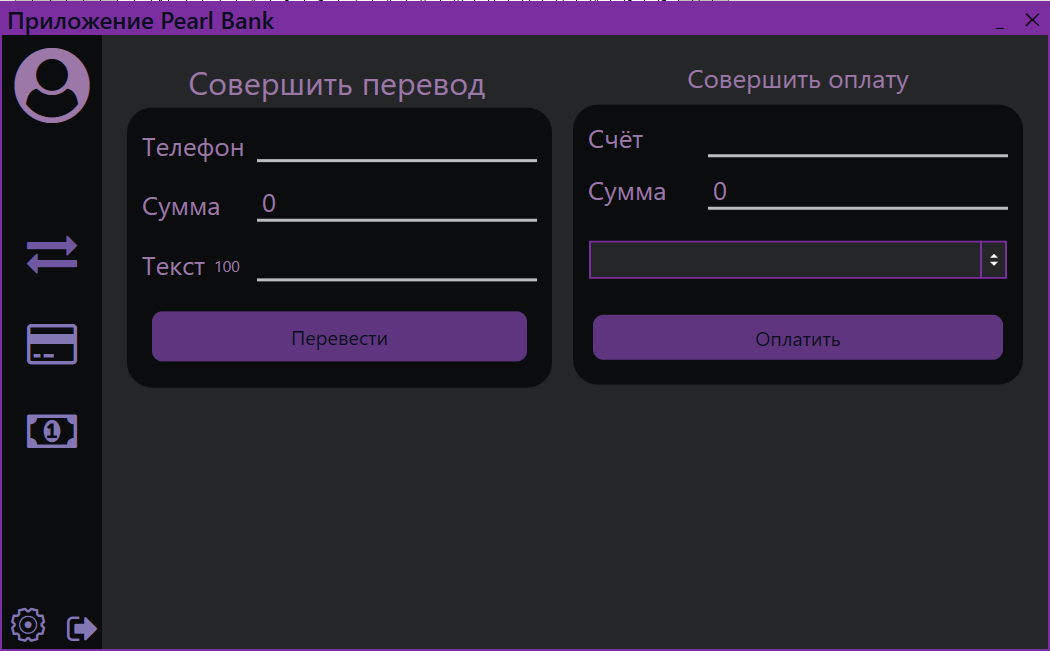


Рисунок 8 – Страница совершения денежных переводов

На этой странице расположены две небольшие формы, подписанные как «Совершить перевод» и «Совершить оплату».

Форма для совершения перевода трёх текстовых полей, необходимых для ввода такой информации о переводе как:

* Номер телефона получателя,
* Сумма перевода,
* Сообщение.

Каждое из полей имеет подпись, с левой стороны от него, которая соответствует данных, которые должно быть туда введены. Рядом с полем «Сообщение» расположен цифровой счётчик, показывающий, сколько символов доступно для написания сообщения. Максимальное значение счётчика – 100 символов.

В нижней части формы находиться кнопка с надписью «Перевести». Эта кнопка неактивна до тех пор, пока поля «Номер телефона» и «Сумма» не будут заполнены. Поле «Сообщение» не является обязательным, и может оставаться пустым. После нажатия этой кнопки, если в базе данных по введённому номер телефона был найден пользователь, и у отправителя достаточно средств для совершения перевода – все поля в форме станут пустыми, что означает что перевод был отправлен успешно. Иначе на экран будет выведено окно с сообщением об ошибке.

В правой части страницы находится форма, отвечающая за совершение оплаты определённых услуг. Форма состоит из двух текстовых полей с надписями «Счёт» и «Сумма», которые отвечают за номер счёта, на который будет отправлена оплата за услугу, и сумму денег, необходимых для оплаты данной услуги. Чуть ниже находится выпадающий список, предлагающий выбрать одну из категорий затрат, в которую будет определён данный перевод. Список доступных категорий:

* Животные,
* Транспорт,
* Коммунальные услуги,
* Медицина,
* Образование,
* Одежда,
* Отдых,
* Техника,
* Еда.

В самом низу формы находится кнопка с надписью «Оплатить». Пока не заполнены поля «Счёт» и «Сумма» и не выбрана категория затрат, эта кнопка будет неактивна. После их заполнения кнопка станет доступна. Если у пользователя достаточно средств для оплаты выбранной им услуги, после нажатия кнопки «Оплатить» текстовые поля станут пусты, что означает, что услуга была оплачено корректно, и данные в базе данных были обновлены, иначе будет вызвано окно с сообщением об ошибке, возникшей в процессе оплаты.

### 2.1.6 Страница просмотра расходов

Нажатие на иконку банковской карты, расположенной на панели навигации, приведёт пользователя на страницу, на которой будет выведена детальная информация о всех расходах и переводах за период использования приложения. Интерфейс страницы показан на рисунке 9.

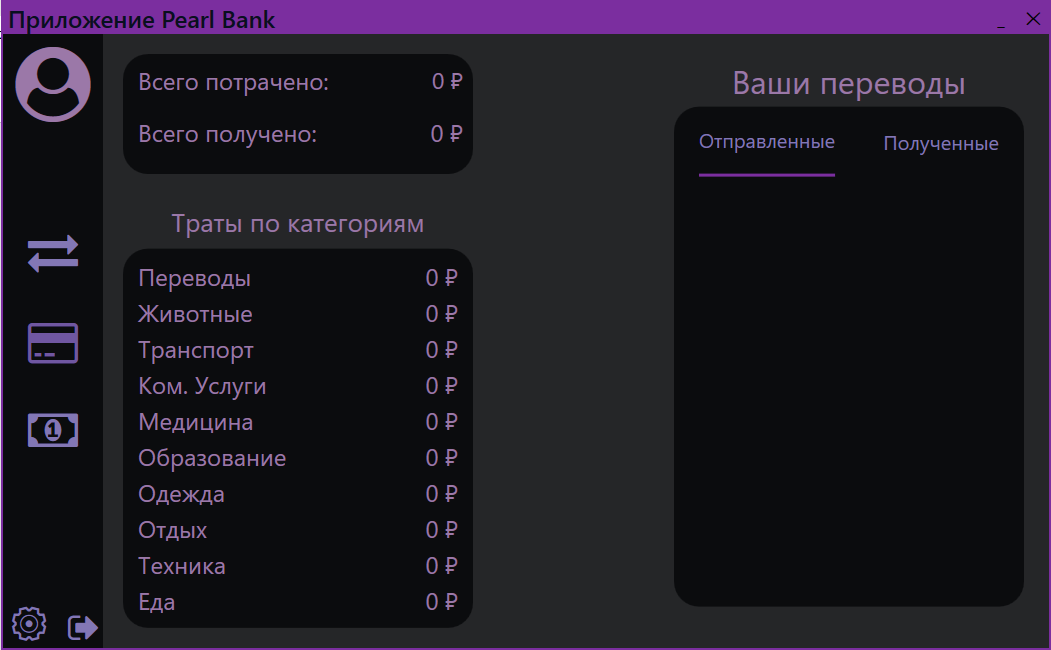


Рисунок 9 – Страница просмотра расходов

В левом верхнем углу этой страницы расположена рамка с двумя строками текста, в которых написано о том, сколько всего денег было потрачено пользователем, и сколько было получено.

Под ней находится рамка с подписью «Траты по категориям», в которой расположен список названий всех доступных в приложении категорий затрат, и количество денег, потраченное на каждую отдельную категорию.

В правой половине страницы находится форма, подписанная как «Ваши переводы». В её верхней части находятся две кнопки – «Отправленные» и «Полученные». Изначально кнопка с надписью «Отправленные» помечена как активная, и под ней, в нижней части формы, находится список всех переводов, которые когда–либо были отправлены пользователем с данного аккаунта. Нажатие на кнопку «Полученные» заменит список переводов на те, которые были получены пользователем с других аккаунтов. Для каждого перевода указаны такие данные как:

* Ф.И.О. Отправителя,
* Ф.И.О. Получателя,
* Сумма перевода,
* Сообщение, если оно было отправлено.

### 2.1.7 Окно предупреждения

При возникновении во время выполнения приложения каких–либо ошибок, на экран будет выведено окно с предупреждением. Это небольших размеров окошко, состоящее из панели, находящейся в верхней его части, на которой слева расположено название ошибки, а справа – кнопка «Закрыть», закрывающая данное окно, и основной части, в которой расположен текст выводимой ошибки и кнопка с надписью «ОК», нажатие на которую подразумевает что пользователь понял, в чём ошибка и постарается исправить её чтобы продолжить пользоваться приложением дальше. Пример окна предупреждения находится на рисунке 10.

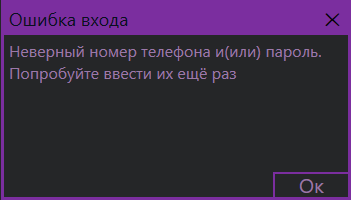


Рисунок 10 – Пример окна предупреждения

## 2.2 Пользовательские сценарии

Пользовательский сценарий, это схема взаимодействия пользователя с приложением. Они могут быть необходимы для того, что лучше понять поведение пользователя в процессе использования приложения, определить возможные недостатки, поэтому составление подобных сценариев очень важно при разработке сложных приложений. В ходе разработки данного приложения были выявлено несколько возможных сценариев.

Чтобы создать аккаунт в приложении, пользователю необходимо:

* Открыть приложение,
* Нажать на кнопку «Создать аккаунт»,
* Ввести в текстовые поля необходимые данные,
* Убедиться, что данные были введены верно,
* Нажать на кнопку «Создать аккаунт».

Для того, чтобы просмотреть полученные переводы, необходимо:

* Открыть приложение,
* Войти в аккаунт,
* Нажать на иконку банковской карты на навигационной панели,
* Нажать на кнопку «Полученные».

Для перевода денег из одной валюты в другую, нужно:

* Открыть приложение,
* Войти в аккаунт,
* Нажать на иконку купюры на навигационной панели,
* Выбрать нужную форму для перевода,
* На форме, в левом текстовом поле, ввести необходимое количество денег для перевода.

# ЗАКЛЮЧЕНИЕ

Следует рассмотреть результаты проделанной работы по разработке приложения для банка, отметить цели, которые были достигнуты и которых можно достигнуть в будущем. Проблемой являлось создание надёжных алгоритмов, которые обеспечат бесперебойную и точную работу системы.

Стоит отметить, что готовое приложение позволяет пользователю зарегистрировать аккаунт, совершать денежные переводы на счета других пользователей, оплату услуг и осуществлять перевод валюты из одной в другую по доступным курсам. Помимо этого, были реализованы возможности изменения цветового оформления интерфейса, данных аккаунта пользователя и пополнения счёта прямо внутри приложения.

Для работы приложения была создана надёжная и удобная система управления информацией из базы данных, в которой хранятся данные о пользователях и их действиях.

Из задач, которые не были достигнуты либо не реализованы должным образом, стоит вынести:

* Оптимизация системы регистрации пользователей,
* Оптимизация системы отслеживания возможных нарушений со стороны пользователей,
* Добавления большего количества настроек в приложение,
* Добавить возможность восстановления либо изменения пароля через приложение,
* Перенос базы данных на удалённый сервер.

В заключении можно отметить, что все основные требования к приложению были выполнены, что означает, что цель работы была достигнута.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

**Учебники и учебные пособия**

1. Прайс. C#7 и .NET Core. Кросс–платформенная разработка для профессионалов, 2018
2. Скит. C# для профессионалов. Тонкости программирования, 2019
3. Албахари. C# 7.0. Справочник. Полное описание языка, 2018
4. Албахари, Албахари: C# 8.0. Карманный справочник, 2020

**Ресурсы сети Интернет**

1. Сайт о программировании язык C# и платформа .NET <https://metanit.com/sharp/> [Электронный ресурс]
2. Руководство по языку C#, краткий обзор язык C# [https://docs.microsoft.com/ru–ru/dotnet/csharp/](https://docs.microsoft.com/ru-ru/dotnet/csharp/) [Электронный ресурс]
3. Электронный учебник по платформе WPF [https://wpf–tutorial.com/](https://wpf-tutorial.com/) [Электронный ресурс]
4. Полный электронный путеводитель по языку C# [https://csharp.net–tutorials.com/](https://csharp.net-tutorials.com/) [Электронный ресурс]
5. C# для начинающих Бесплатный курс по C# [https://ru.code–basics.com/languages/csharp](https://ru.code-basics.com/languages/csharp) [Электронный ресурс]
6. Stackoverflow – вопросы и ответы для программистов <https://ru.stackoverflow.com/> [Электронный ресурс]

**Лист нормоконтроля**

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

(подпись)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Нормоконтроль

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

(подпись) (и.о. фамилия)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.