Министерство образования Республики Беларусь

Учреждения образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ

И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №4

на тему

**ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

Студент гр. 053503 Карачун Д. Ю.

Преподаватель Тушинская Е. В.

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc152958663)

[1 Описание программного продукта 4](#_Toc152958667)

[2 Тестирование приложения 9](#_Toc152958668)

[3 Развертывание приложения 11](#_Toc152958669)

[Заключение 13](#_Toc152958670)

[Приложение А Листинг кода тестирования 14](#_Toc152958671)

# **ВВЕДЕНИЕ**

# Современный мир невозможно представить без финансов: валютно-фондовые биржи, банковские системы, активный рост криптовалюты – всё это мотивирует разработчиков создавать более прогрессивные и надежные финансовые приложения и технологии. Финтех как сфера деятельности программистов набирает всё большую популярность за счёт появления спроса на соответствующее программное обеспечение.

# Наиболее часто используемыми сервисами финтех-индустрии являются интернет-банкинги, которые позволяют пользователям совершать различные финансовые операции, такие как переводы, платежи, открытие вкладов, оформление кредитов и т.д. Исходя из этого факта, можно предположить, что и потребительский спрос на такие приложения высок.

# В процессе изучения индустрии для её дальнейшего развития было принято решение создать программный продукт, целью которого является разработка сервиса с надежной банковской системой для осуществления и обработки платежей.

Таким образом, целью данного проекта является разработка банковской системы, реализующая все функции и процессы, необходимые клиенту банка для осуществления и контроля платежей.

# **1 ОПИСАНИЕ ПРОГРАММНОГО ПРОДУКТА**

К последней лабораторной работе было разработано свое банковскоеприложение, которое имеет весь базовый функционал аналогов данной предметной области.

Для начала приложение устанавливается установщиком с ftp-сервера. Затем готовый .exe-файл можно использовать на компьютере. При открытии его нас встречает страница входа, что видно на рисунке 1.1.

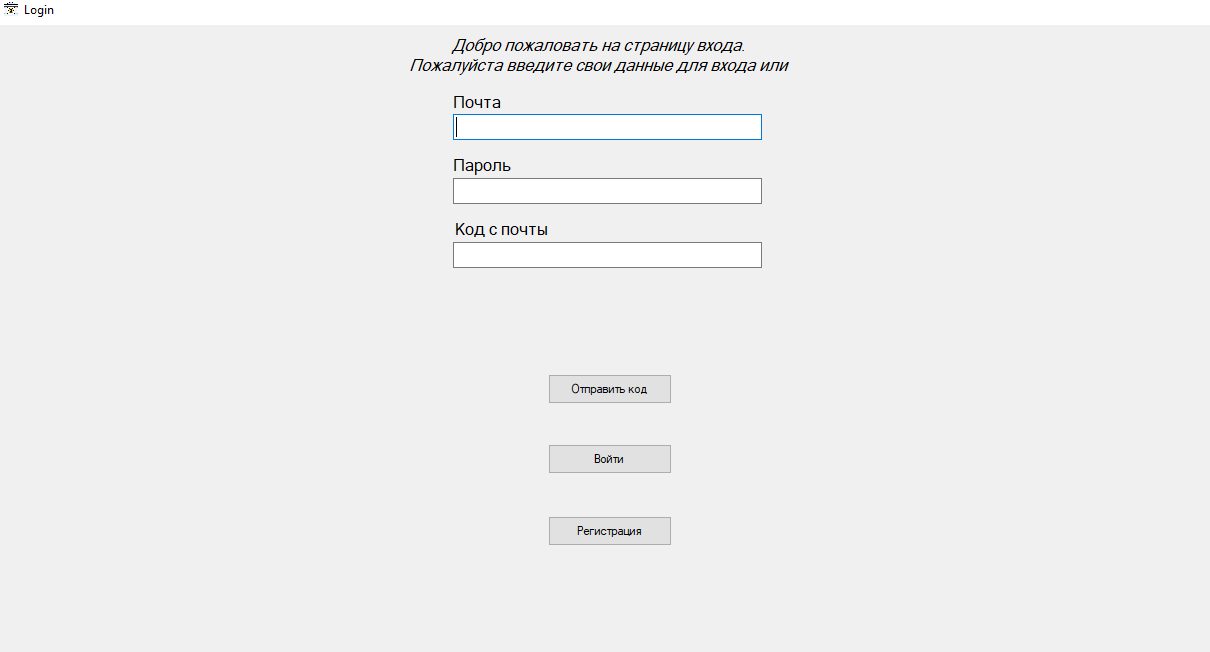


Рисунок 1.1 – Страница входа

Если же пользователь не зарегистрирован, то нажатием на кнопку “Регистрация” его перебросит на страницу регистрации, вид которой представлен на рисунке 1.2.

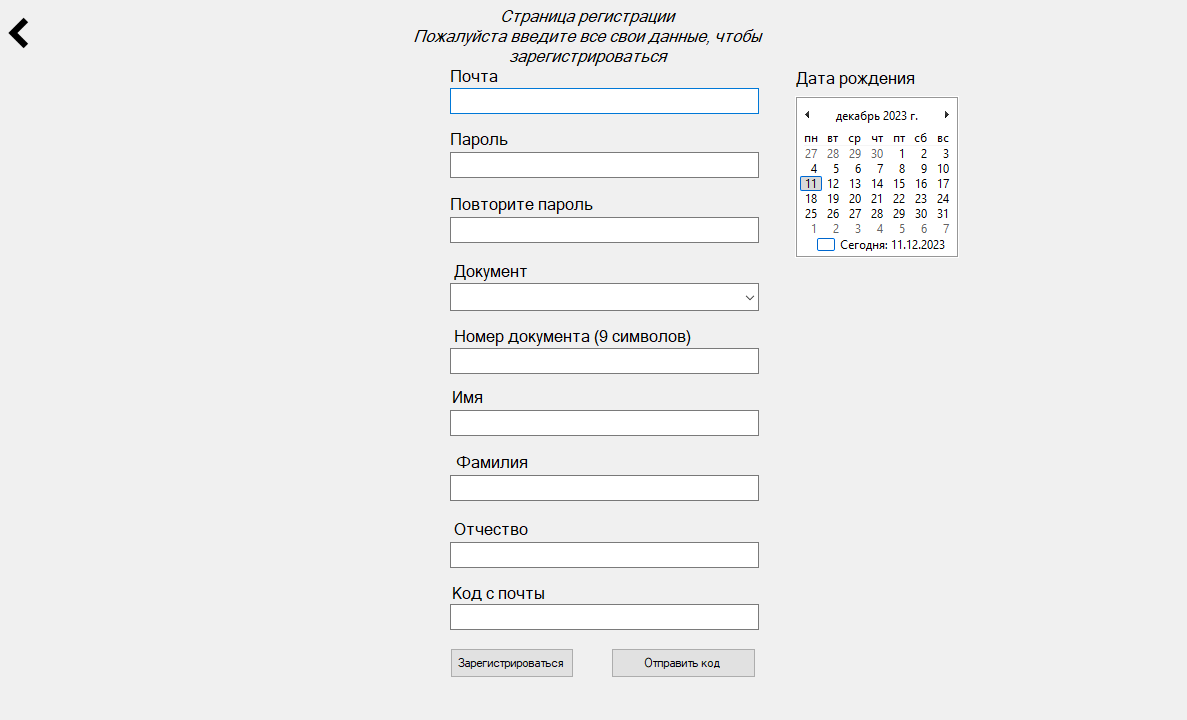


Рисунок 1.2 – Окно регистрации

Регистрационная панель была оформлена с учетом требований, которые были описаны в технической записке проекта.

После входа в приложение нас встречает окно с набором функций для выбора, на рисунке 1.3

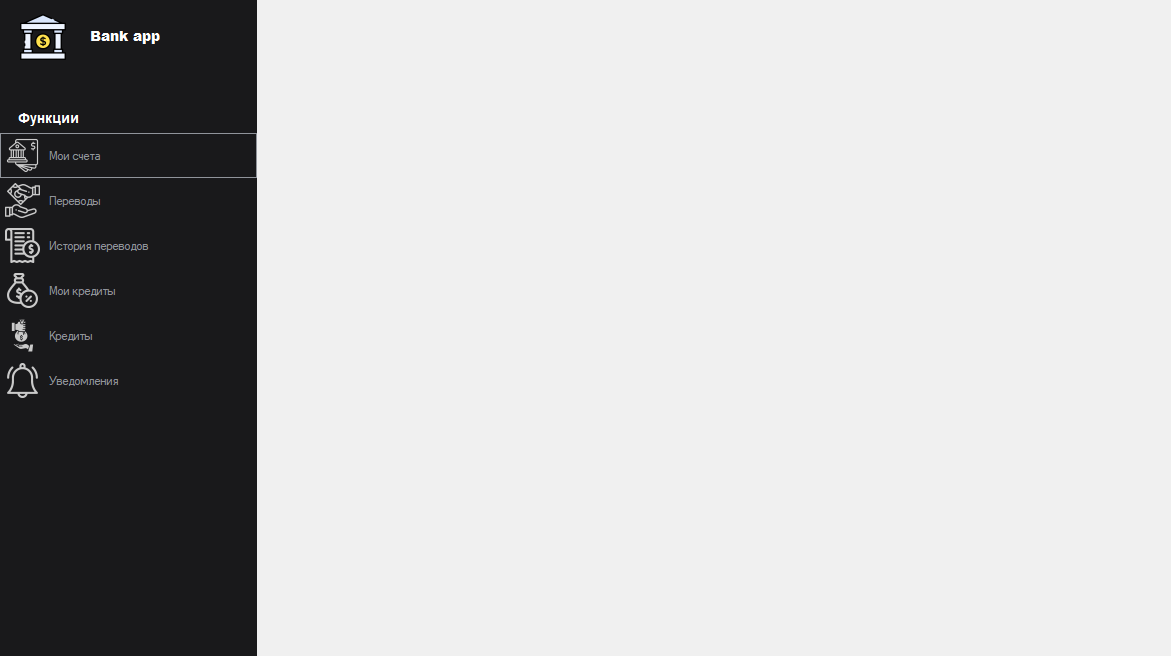


Рисунок 1.3 – Окно после входа

Здесь мы может выбрать что нас интересует. Например страница моих счетов, на рисунке 1.4

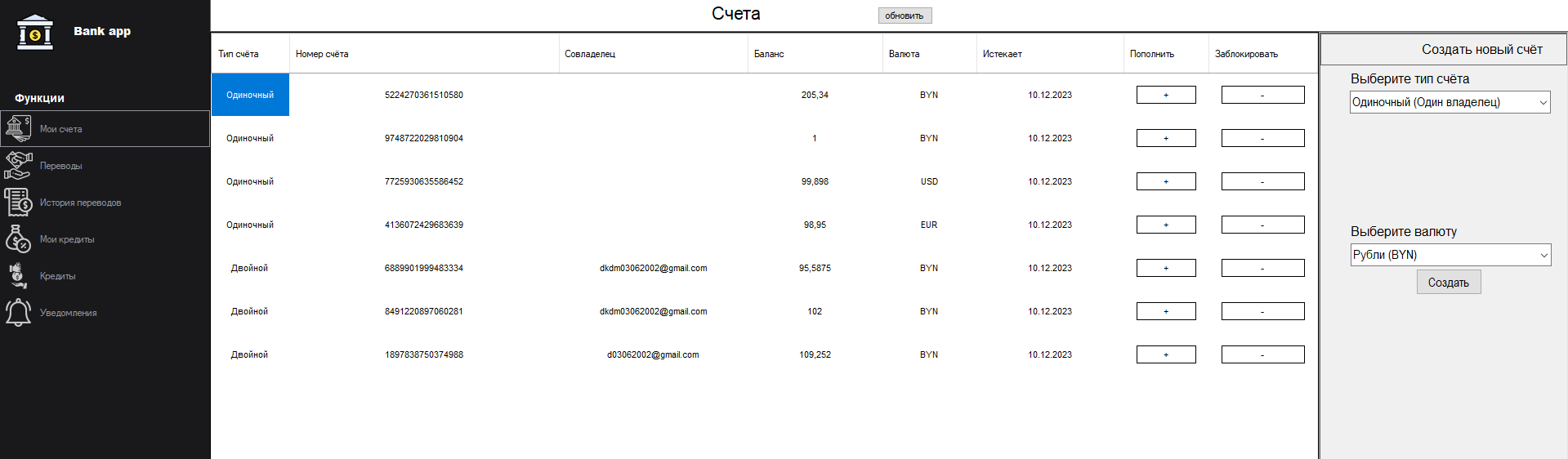


Рисунок 1.4 – Окно счетов

На данной вкладке пользователь может создать новый счёт с одной из трёх валют на выбор, а так же создать двойной счёт с разными типами защиты, как это было указано в техническом задании, помимо этого можно просимулировать операцию пополнения счёта и заблокировать не нужный счёт.

Далее можно рассмотреть окно переводов на рисунке 1.5.

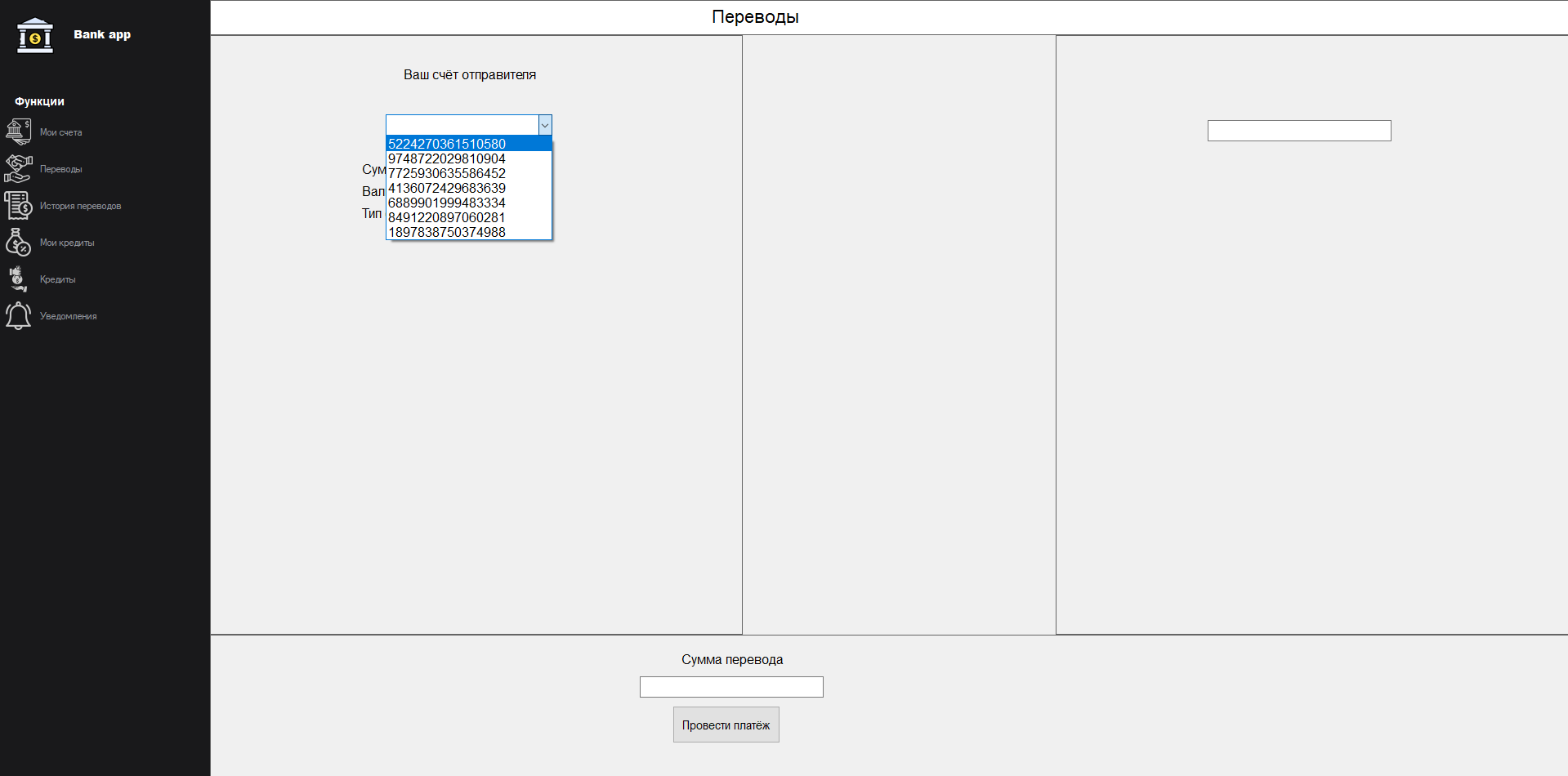


Рисунок 1.5 – Окно переводов

Тут пользователь может выбрать счёт для перевода, сумму и счёт-получатель денежный средств. Если же счета имеют разные валюты, то происходит конвертация по актуальному курсу с комиссией в пять процентов.

Далее посетим окно истории переводов на рисунке 1.6.



Рисунок 1.5 – Окно истории транзакций

Здесь перечислены все финансовый операции со счетами, которые принадлежат пользователю. От простых переводов, то зачислений и оплат кредитов.

Перейдём на вкладку пользовательских кредитов, которая представлена на рисунке 1.6.



Рисунок 1.6 – Окно пользовательских кредитов

В данном окне пользователь может просмотреть свои активные и закрытые кредиты, информацию о ежемесячном платеже и непосредственно провести платёж.

Посети окно всех возможных кредитов, оно представлено на рисунке 1.7.

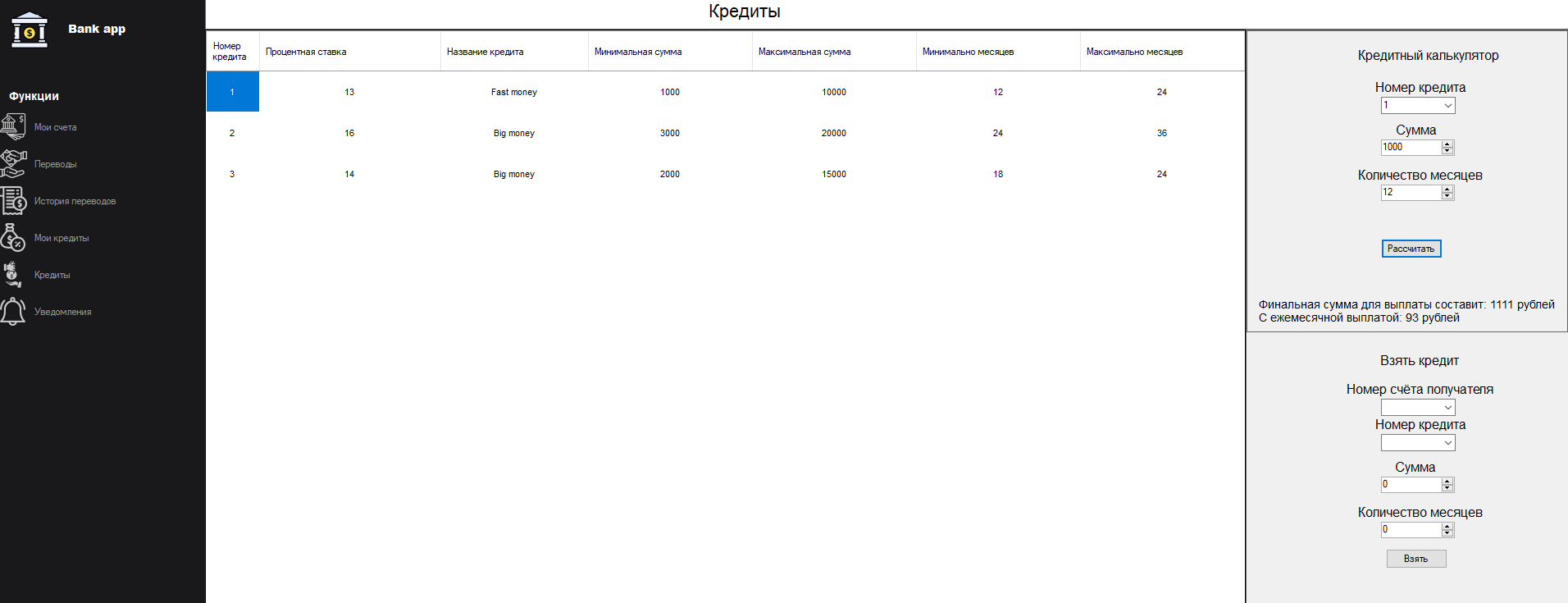


Рисунок 1.7 – Окно всех кредитов

В данном окне пользователь может просмотреть все возможные кредиты, рассчитать конечную сумму и ежемесячные выплаты используя кредитный калькулятор, а так же взять желаемый кредит.

# **2 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ**

Для тестирование программного обеспечения будет производиться юнит-тестирование. Юнит-тесты формируют основу для проведения регрессионного тестирования, гарантируя что система будет вести себя согласно сценарию, когда добавятся новые функциональные возможности или изменятся существующие.

С помощью написанных тестов проверяется правильность работы базы данных и запросов к ней, так же проверяются возможности пользователя и реакция приложения на различные ошибки и неправильно введённые пользователем данные. Так, на рисунке 2.2 представлен вид неправильных данных пользователе при попытке входа, а на рисунке 2.3 представлены указания на неправильный ввод данных для регистрации. Помимо отображения красным текстом, в основном приложении присутствуют и всплывающие сообщения о неправильно введённых данных.

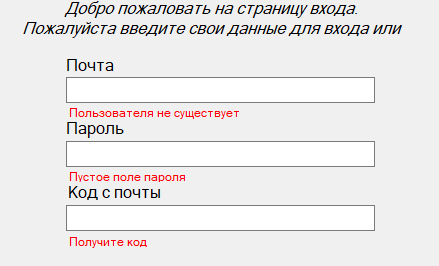


Рисунок 2.1 – Окно при неправильной попытке входа

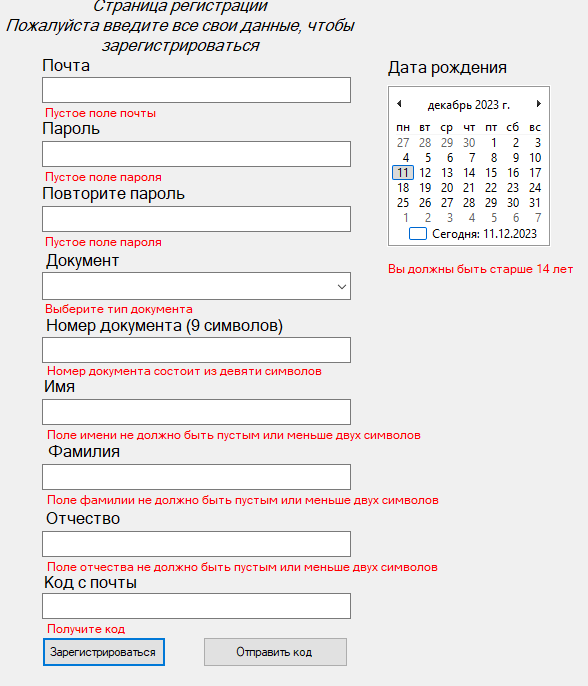


Рисунок 2.2 – Окно при неправильной попытке регистрации

Таким образом, в результате юнит-тестов и ручных тестов ошибок в работе приложении обнаружено не было и все формы грамотно отработали приём данных от пользователя. База данных на все запросы реагирует корректно и предоставляет нужные данные.

Листинг кода тестирования представлен в приложении А.

# **3 РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЯ**

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их. Включает в себя следующие элементы: узлы, артефакты, соединения, устройства.

На рисунке 3.1 приведена диаграмма развёртывания для разработанного приложения.

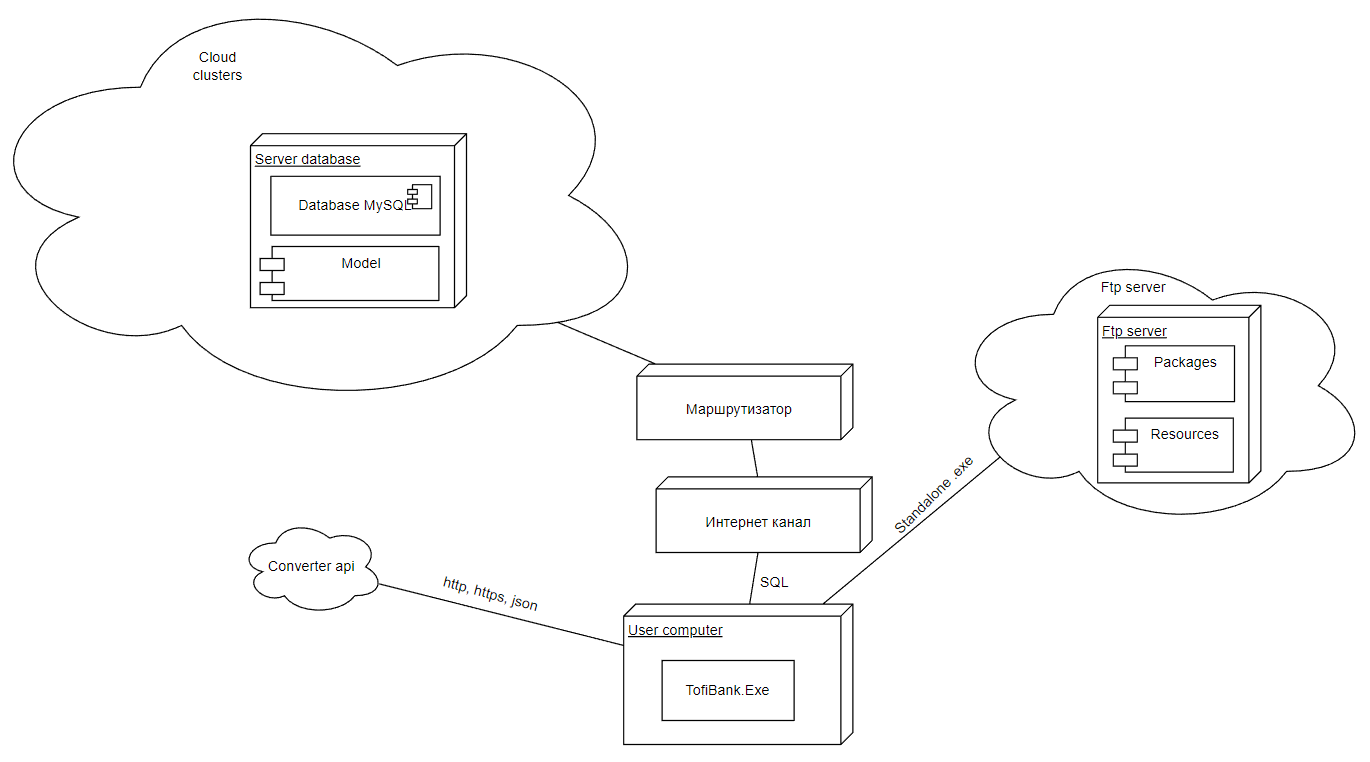


Рисунок 3.1 – Диаграмма развертывания

Здесь, представлен облачный сервер для базы данных, FTP-сервер для хранения файлов самого приложение и генерации и загрузки exe-файла банкинга на компьютер пользователя. Так же приложение с компьютера пользователя связывается с облачным конвертером валют, для получения актуальных курсов валют.

**3.1 Развертывание MySQL**

Для развёртывания MySQL базы данных был выбран сервис CloudClusters. В нём можно достаточно быстро и выгодно получить готовый сервер с базой данных и затем, в пару кликов настроить приложения для работы с данным удалённым сервером.

Сам exe-файл с различными пакетами для его работы был загружен на FTP-сервер, доступ к которому пользователь получает из файла установщика и затем, данный установщик подтягивает с сервера уже готовый для работы и пользования exe-файл. Так же данный сервер поддерживает различные версии и в случае, если приложение было обновлено, на компьютер пользователя подтянется уже новая версия приложения.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения был разработан полноценный программный продукт упрощённого аналога банковской системы, а также написаны тесты для тестирования ключевой логики приложения. В качестве дополнительных видов тестирования для приложения также можно реализовывать:

– функциональное тестирование компонентов;

– тестирование пользовательских сценариев;

База данных приложения развёрнута на удалённом сервере, а само приложение скачивается установщиком с другого сервера. Также разработана диаграмма развёртывания и составлен отчёт, оформленный в соответствии с общими требования стандарта предприятия БГУИР.

В приложении А приведена вырезка кода из тестирования базы данных, которая совершает действия перед началом тестирования и после окончания тестирования.

Цели лабораторной работы можно считать достигнутыми.

# **ПРИЛОЖЕНИЕ А**

# **(обязательное)**

# **Листинг кода тестирования**

[TestClass]

public class Testing

{

[TestMethod]

public void TestBalanceTransactiion1()

{

double balance = TestingClass.getTransactionBal(Convert.ToString(1));

Assert.AreEqual(100, balance);

}

[TestMethod]

public void TestBalanceTransactiion2()

{

double balance = TestingClass.getTransactionBal(Convert.ToString(2));

Assert.AreEqual(100, balance);

}

[TestMethod]

public void TestBalanceTransactiion3()

{

double balance = TestingClass.getTransactionBal(Convert.ToString(3));

Assert.AreEqual(1, balance);

}

[TestMethod]

public void CheckUserExists1()

{

bool exists = TestingClass.checkUserExistence("d03062002@gmail.com", "denis38411");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckUserExists2()

{

bool exists = TestingClass.checkUserExistence("dkdm03062002@gmail.com", "denis38411");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckAccountExists1()

{

bool exists = TestingClass.checkAccountExistenceS("5224270361510580");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckAccountExists2()

{

bool exists = TestingClass.checkAccountExistenceS("7725930635586452");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckAccountExists3()

{

bool exists = TestingClass.checkAccountExistenceS("4136072429683639");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckAccountIsNotAvailable1()

{

bool exists = TestingClass.checkAccountExistenceS("2613561378622809");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckAccountIsNotAvailable2()

{

bool exists = TestingClass.checkAccountExistenceS("-207198988");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckDocumentExistence1()

{

bool exists = TestingClass.checkDocumentExistence("MP3824468");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckDocumentExistence2()

{

bool exists = TestingClass.checkDocumentExistence("MP3824469");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckEmailExistence1()

{

bool exists = TestingClass.checkEmailExistence("d03062002@gmail.com");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckEmailExistence2()

{

bool exists = TestingClass.checkEmailExistence("dkdm03062002@gmail.com");

Assert.IsTrue(exists);

}

[TestMethod]

public void CheckEmailIsFree()

{

bool exists = TestingClass.checkEmailExistence("drmdm@gmail.com");

Assert.IsFalse(exists);

}

[TestMethod]

public void CheckEmailIsFree2()

{

bool exists = TestingClass.checkEmailExistence("freemail@gmail.com");

Assert.IsFalse(exists);

}

[TestMethod]

public void CheckUserID1()

{

int id = TestingClass.getUserID("dkdm03062002@gmail.com");

Assert.AreEqual(2, id);

}

[TestMethod]

public void CheckUserID2()

{

int id = TestingClass.getUserID("d03062002@gmail.com");

Assert.AreEqual(1, id);

}

}