**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/6 группы

Нестерович Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «09» января 2023 г. по «04» февраля 2023 г.

Руководитель практики Челищева Л.Д.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/6 группы

Нестерович Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «09» января 2023 г. по «04» февраля 2023 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения;
* Разработка модулей программного обеспечения;
* Тестирование программных модулей и их интеграции;
* Разработка программной документации и стандарты кодирования.

**Индивидуальное задание: ВАРИАНТ 16**

Задание выдал «09» января 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Челищева Л.Д.

(подпись) (Ф.И.О.)

Задание получил «09» января 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_ Нестерович Н.Д. (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/6 группы

Нестерович Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «09» января 2023 г. по «04» февраля 2023 г.

Руководитель практики Челищева Л.Д.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2023

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 09.01.23 | Анализ предметной области |  |
| 10.01.23 | Выявление требований к программе |  |
| 11.01.23 | Разработка технического задания |  |
| 12.01.23 | UML. Проектирование диаграммы вариантов использования |  |
| 13.01.23 | UML. Проектирование диаграммы последовательности |  |
| 14.01.23 | UML. Проектирование диаграммы активности |  |
| 16.01.23 | Моделирование структуры ПО |  |
| 17.01.23 | Проектирование инфологической и даталогической модели данных |  |
| 18.01.23 | Проектирование интерфейса пользователя. Создание Wireframe эскизов |  |
| 19.01.23 | Разработка дизайна программы в соответствии с руководством по стилю |  |
| 20.01.23 | Разработка базы данных |  |
| 21.01.23 | Разработка словаря данных |  |
| 23.01.23 | Создание приложения. Форма авторизации |  |
| 24.01.23 | Создание приложения. Форма заказов |  |
| 25.01.23 | Создание приложения. Основные формы приложения |  |
| 26.01.23 | Разработка библиотеки классов |  |
| 27.01.23 | Подготовка отчетов и выгрузка документов для печати |  |
| 28.01.23 | Отладка программных модулей |  |
| 30.01.23 | Модульное тестирование |  |
| 31.01.23 | Создание тестовых случаев |  |
| 01.02.23 | Интеграционное тестирование |  |
| 02.02.23 | Разработка самодокументирующегося кода |  |
| 03.02.23 | Инспекция кода на соответствие стандартам кодирования |  |
| 04.02.23 | Подготовка отчета и размещение результатов в репозитории контроля версий |  |

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc126356766)

[1. Проектирование системы 7](#_Toc126356767)

[1.1 ER-диаграмма 7](#_Toc126356768)

[1.2 Диаграмма вариантов использования 10](#_Toc126356769)

[1.3 Диаграмма активности 11](#_Toc126356770)

[1.3 Диаграмма последовательности 12](#_Toc126356771)

[2. Создание базы данных 13](#_Toc126356772)

[3. Разработка программы 14](#_Toc126356773)

[3.1 Создание формы авторизации и главной формы 14](#_Toc126356774)

[3.2 Разработка функционала администратора 17](#_Toc126356775)

[3.3 Разработка функционала менеджера 21](#_Toc126356776)

[3.4 Разработка функционала бухгалтера 24](#_Toc126356777)

[3.5 Разработка библиотеки классов 28](#_Toc126356778)

[3.6 Текст программы 29](#_Toc126356779)

[4. Тестирование программы 30](#_Toc126356780)

[Заключение 31](#_Toc126356781)

[Приложение А 32](#_Toc126356782)

[Приложение В 37](#_Toc126356783)

[Приложение Г 57](#_Toc126356784)

Введение

Целью учебной практики являлось изучение и закрепление знаний поэтапной разработки приложения как продукции производственно-технического назначения. Для этого была поставлена задача по проектировке, разработке и тестирования автоматизированной информационной системы.

1. Проектирование системы

1.1 ER-диаграмма

Для иллюстрации отношений и связей между ними была разработана ER-диаграмма (рисунок 1), которая удовлетворяет требованиям для создания приложения, а также словарь данных. Словарь данных представлен в таблицах 1–6.



Рисунок 1 – ER диаграмма

Роли – таблица ролей, хранящая их названия.

Пользователи – таблица пользователей, в которой хранится логин и пароль, а также ID роли пользователей.

Банки - таблица банков, в которой хранится их название и юридический адрес.

Клиенты – таблица клиентов, хранящая в себе их имя и адрес, а также номер его карты и ID банка, выпустившего карту.

Банкоматы – таблица банкоматов, хранящая в себе их адрес, а также ID банка, которому принадлежит банкомат.

Операции – таблица операций, хранящая в себе информацию о клиенте, обналичивающем деньги, банкомате, который выдал деньги, а также сумму снятия и дату операции.

Таблица 1 – Роли

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Сотрудники** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | Autoincrement, not null |
|  | Название | NVARCHAR (50) | N |  |
|  | is\_archived | BIT | N |  |

Таблица 2 – Пользователи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Товары** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | Autoincrement, not null |
|  | Логин | NVARCHAR (30) | N |  |
|  | Пароль | NVARCHAR (30) | N |  |
| FK | ID роли | INTEGER | Y | References Роли (ID) |
|  | is\_archived | BIT | N |  |

Таблица 3 – Банки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Должности** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | Autoincrement, not null |
|  | Название | NVARCHAR (50) | N |  |
|  | Юридический адрес | NVARCHAR (150) | N |  |
|  | Is\_archived | BIT | N |  |

Таблица 4 – Клиенты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Фирмы** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | Autoincrement, not null |
|  | ФИО | NVARCHAR (50) | N |  |
|  | Номер карты | BIGINT | N |  |
|  | Адрес | NVARCHAR (150) | N |  |
| FK | ID банка | INTEGER | Y | References Банки (ID) |
|  | Is\_archived | BIT | N |  |

Таблица 5 – Банкоматы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Заказы** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | Autoincrement, not null |
|  | Адрес | NVARCHAR (150) | N |  |
| FK | ID банка | INTEGER | Y | References Банки (ID) |
|  | Is\_archived | BIT | N |  |

Таблица 6 - Операции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Исполненные** | | | | |
| **KEY** | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | ID | INTEGER | Y | unique, not null |
| FK | ID клиента | INTEGER | Y | References Клиенты (ID) |
| FK | ID банкомата | INTEGER | Y | References Банкоматы (ID) |
|  | Дата\_время | DATE | N |  |
|  | Сумма | MONEY | N |  |
|  | Is\_archived | BIT | N |  |

1.2 Диаграмма вариантов использования

Для частичного описания системы была создана диаграмма прецедентов, иллюстрирующая роли системы, а также функционал программы, к которому соответствующие роли имеют доступ (рисунок 2).



Рисунок 2 – Диаграмма вариантов использования (прецедентов)

1.3 Диаграмма активности

На рисунке 3 изображена диаграмма активностей менеджера, приложения и базы данных при добавлении нового клиента.



Рисунок 3 – Диаграмма активности

1.3 Диаграмма последовательности

На рисунке 4 отображена диаграмма последовательности, иллюстрирующая добавление нового клиента менеджером, а также последовательность действий, характеризующих данный процесс.



Рисунок 4 – Диаграмма последовательности

1. Создание базы данных

С помощью СУБД Miscrosoft SQL Server MS была разработана база данных, соответствующая ER-диаграмме и словарю данных. С помощью SQL-кода были созданы 6 таблиц, а также связи между ними.

Подробный листинг SQL-кода представлен в приложении Г.

1. Разработка программы

3.1 Создание формы авторизации и главной формы

При запуске программы, пользователя встречает главная форма (рисунок 5). Чтобы авторизоваться, пользователю нужно ввести логин и пароль и нажать кнопку «Войти».

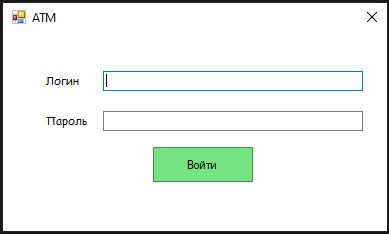


Рисунок 5 – Форма авторизации

При вводе неверного логина и пароля появляется сообщение «Неверный логин или пароль» (рисунок 6).

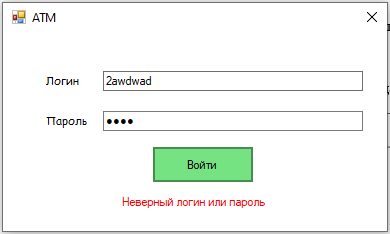


Рисунок 6 – Неверный логин или пароль

При наведении курсора на поле ввода пароля он становится видимым (рисунок 7).

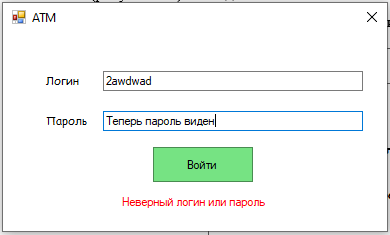


Рисунок 7 – Видимый пароль

При успешной авторизации открывается главная форма соответствующей роли (рисунки 8-10).

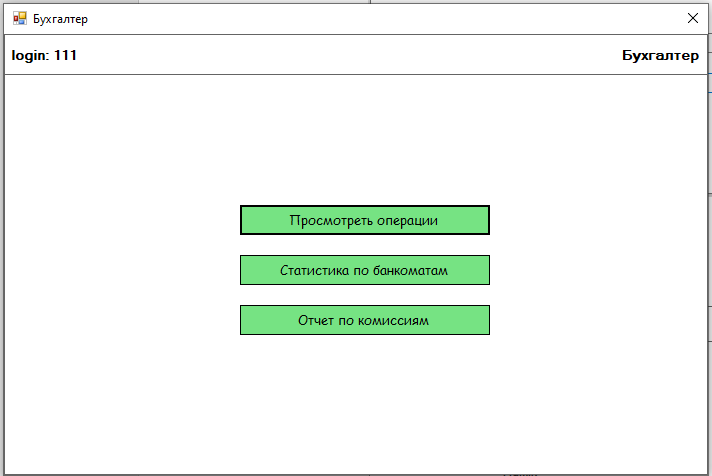


Рисунок 8 – Главная форма администратора

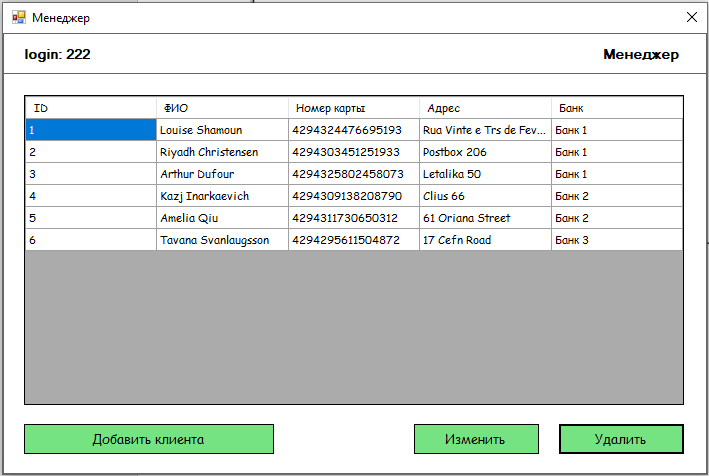


Рисунок 9 – Главная форма менеджера

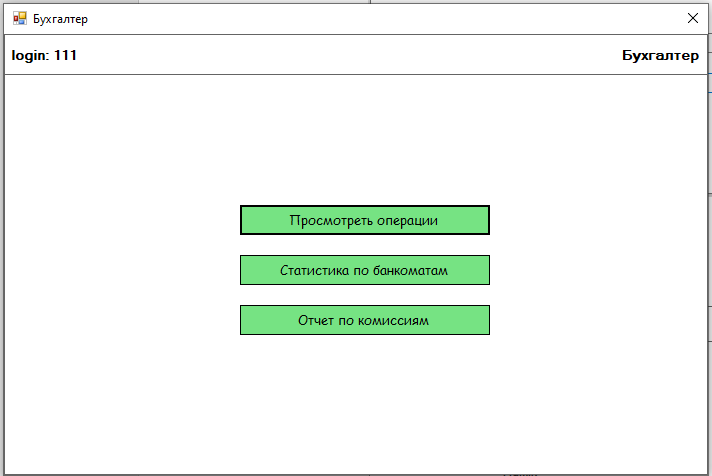


Рисунок 10 – Главная форма бухгалтера

3.2 Разработка функционала администратора

Администратор имеет доступ к просмотру, добавлению, редактированию и удалению банков и банкоматов (рисунки 11-20).

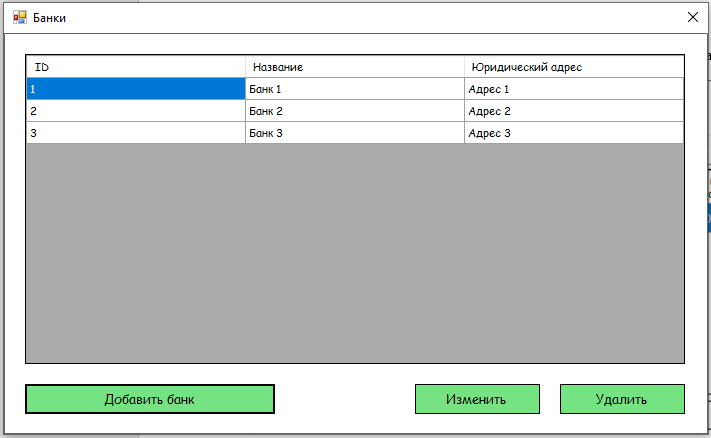


Рисунок 11 – Таблица «Банки»

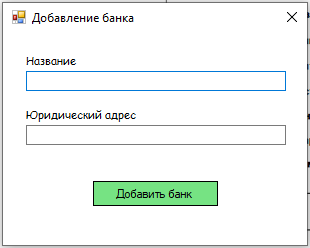


Рисунок 12 – Форма добавления банка

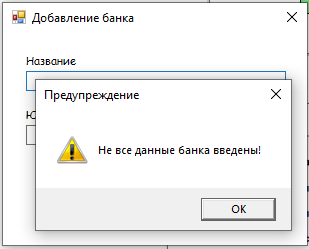


Рисунок 13 – Ошибка при заполнении не всех полей банка

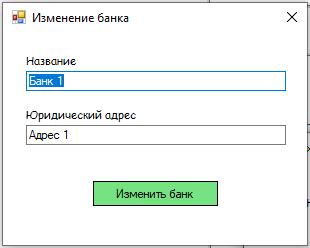


Рисунок 14 – Форма изменения банка

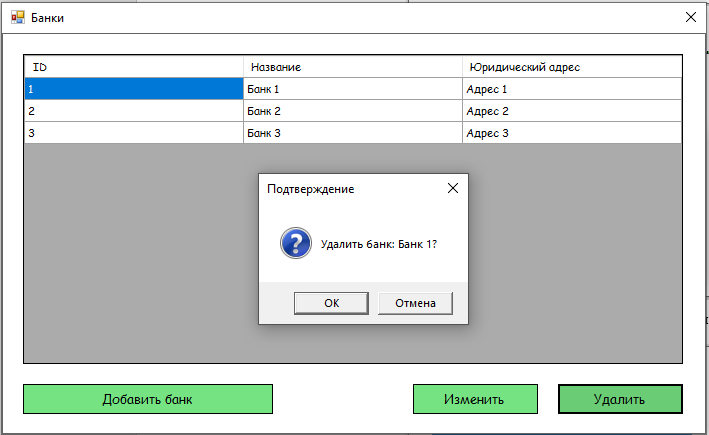


Рисунок 15 – Подтверждение удаления банка

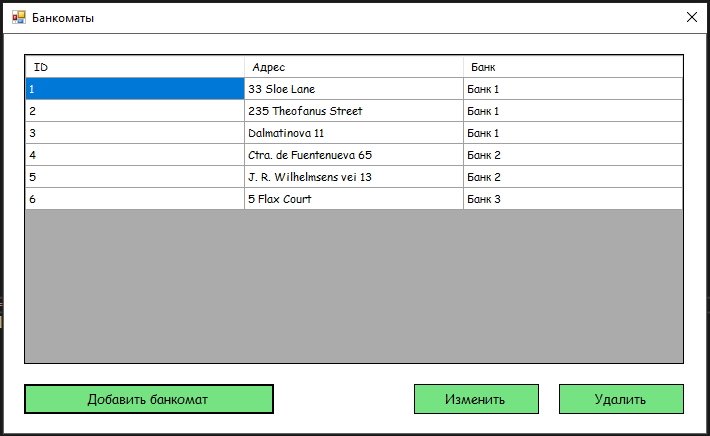


Рисунок 16 – Таблица «Банкоматы»

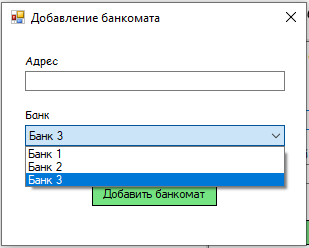


Рисунок 17 – Форма добавления банкомата

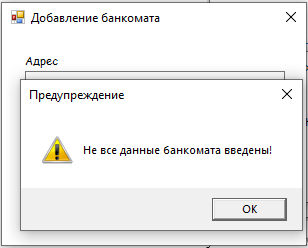


Рисунок 18 – Ошибка при заполнении не всех полей банкомата

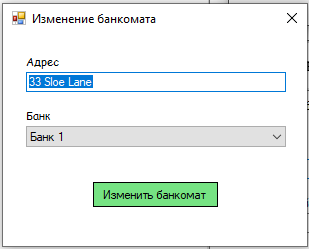


Рисунок 19 – Форма изменения банкомата

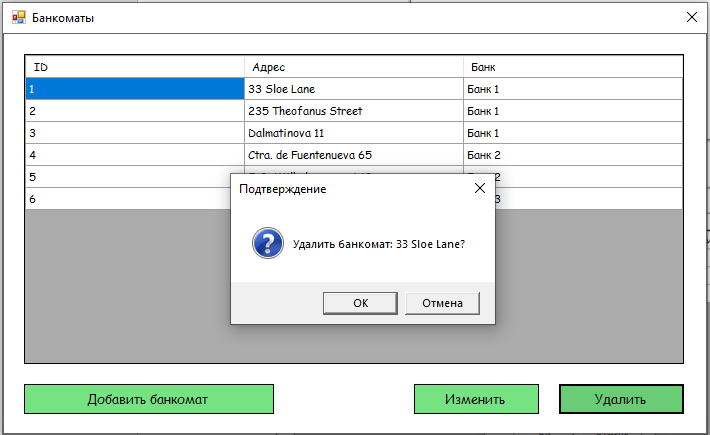


Рисунок 20 – Подтверждение удаления банкомата

3.3 Разработка функционала менеджера

Менеджер имеет доступ к просмотру, добавлению, редактированию и удалению клиентов (рисунки 21-25).

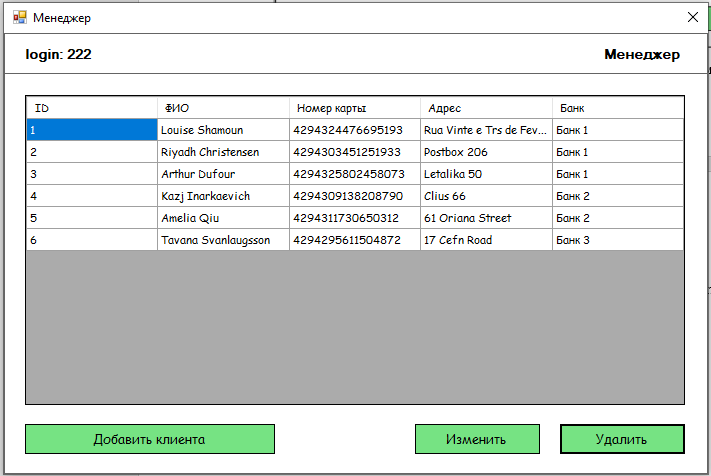


Рисунок 21 – Таблица «Клиенты»

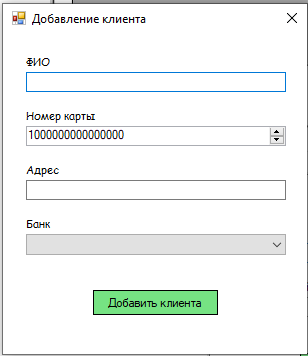


Рисунок 22 – Форма добавления клиента

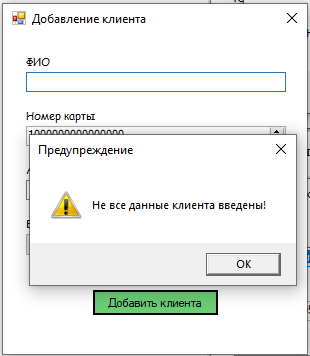


Рисунок 23 – Ошибка при заполнении не всех полей клиента

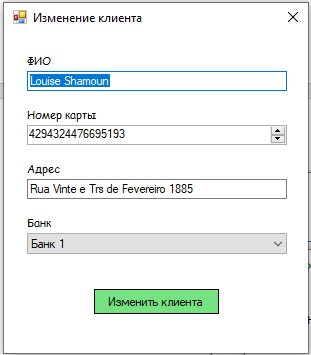


Рисунок 24 – Форма изменения клиента

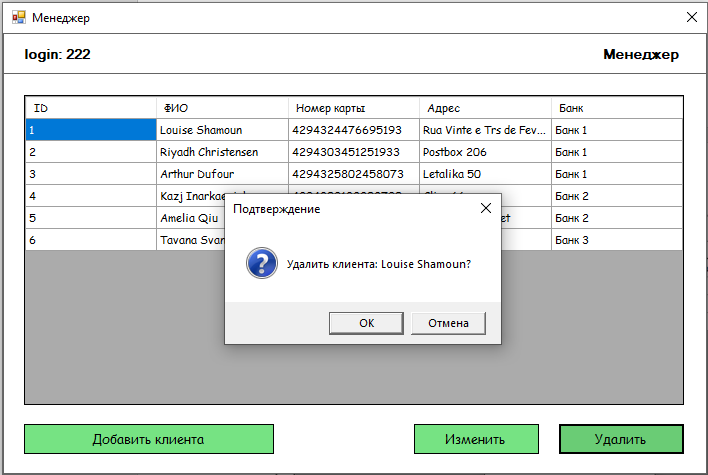


Рисунок 25 – Подтверждение удаления клиента

3.4 Разработка функционала бухгалтера

Бухгалтер имеет доступ к просмотру и сохранению списка операций, а также статистики использования банкоматов и отчета по комиссиям (рисунки 26-28).

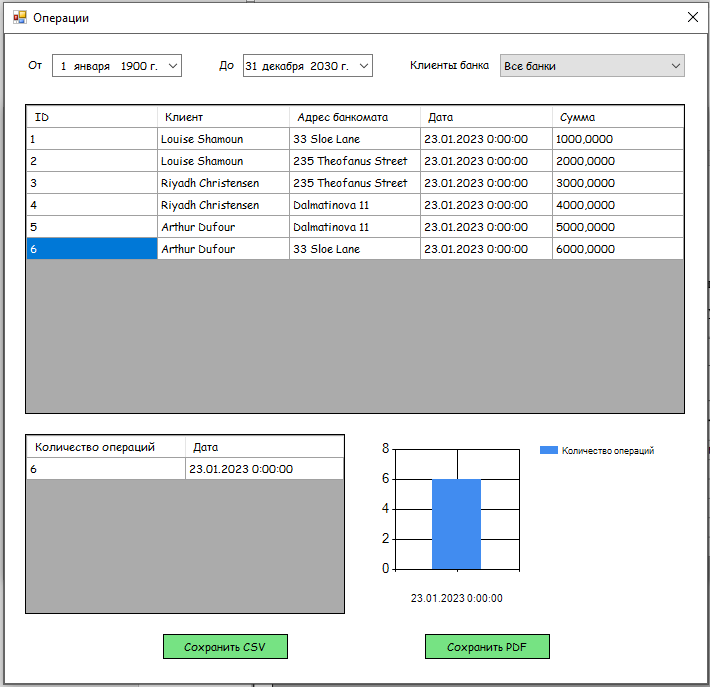


Рисунок 26 – Список операций

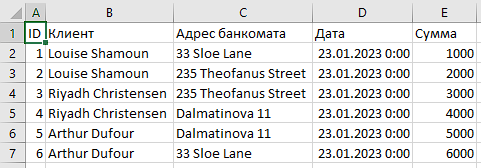


Рисунок 27 – Выгруженный CSV отчет по операциям

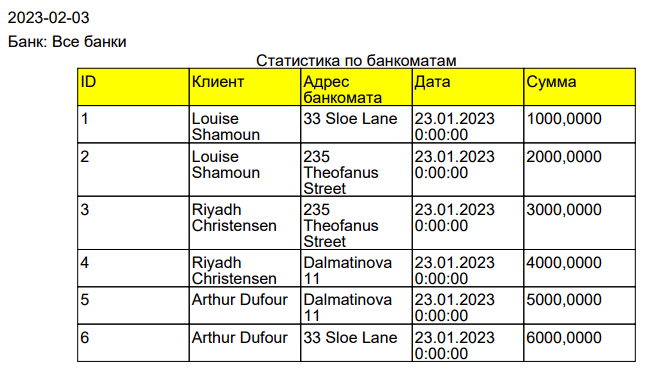


Рисунок 28 – Выгруженный PDF отчет по операциям

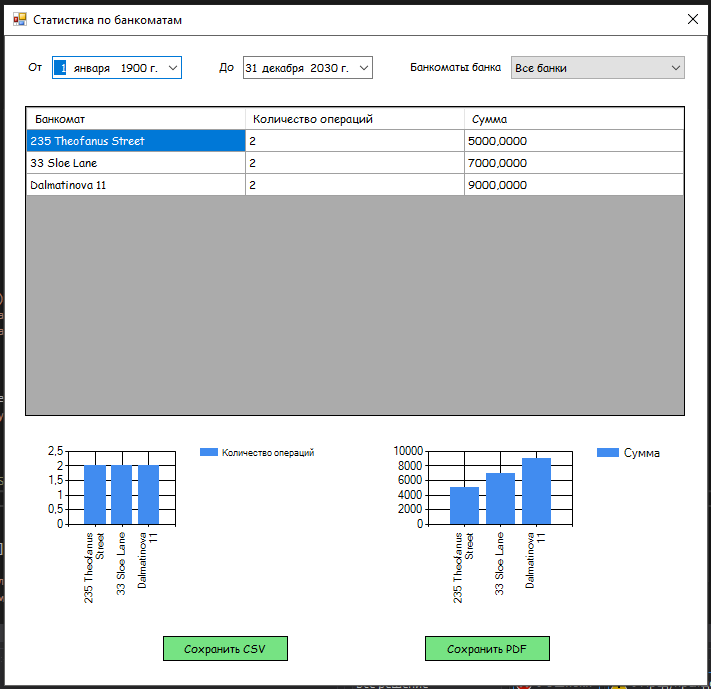


Рисунок 29 – Статистика использования банкоматов

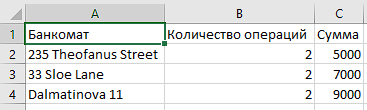


Рисунок 30 – Выгруженный CSV отчет по банкоматам

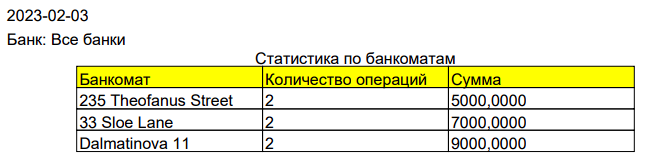


Рисунок 31 – Выгруженный PDF отчет по банкоматам

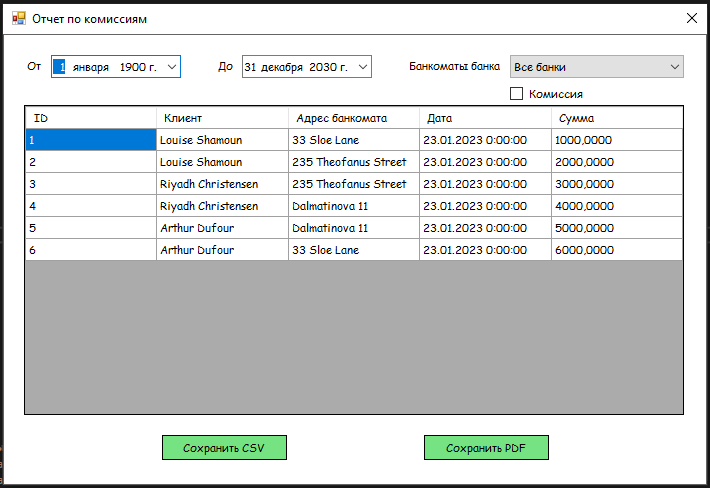


Рисунок 32 – Отчет по комиссиям



Рисунок 33 – Выгруженный CSV отчет по комиссиям

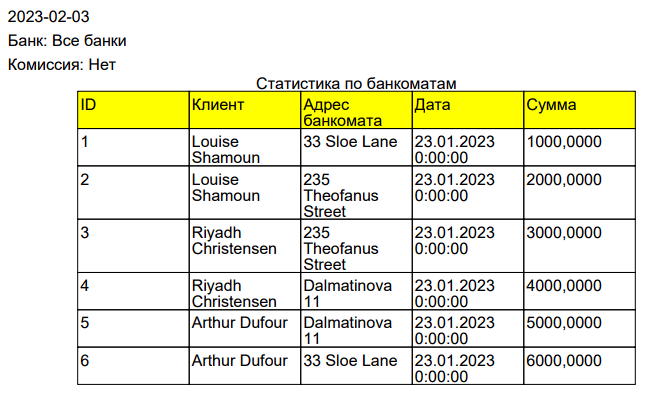


Рисунок 34 – Выгруженный PDF отчет по банкоматам

3.5 Разработка библиотеки классов

Была разработана библиотека классов, представленная в виде .dll файла и подключаемая к основному проекту. Библиотека содержит методы для заполнения элементов форм приложения: ComboBox, DataGridView; метод построения диаграмм; методы выгрузки отчетов в .pdf и .csv файлы.

Код библиотеки представлен в приложении В, после кода приложения.

3.6 Текст программы

Текст программы в соответствии с ГОСТ 19.101–77 (СТ СЭВ 1626–79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

1. Тестирование программы

В ходе разработки программы также были реализованы и пройдены тест кейсы. Тестовые случаи и их результаты отображены в приложении А.

Заключение

Приложение А

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | ATM |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Нестерович Николай |
| **Дата(ы) теста** | 01.02.23 |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет**  **Тестирования** | Низкий |
| **Заголовок/название теста** | Проверка авторизации с верно введенными логином и паролем |
| **Краткое изложение теста** | Форма авторизации закрывается, открывается главная форма соответствующей роли |
| **Этапы теста** | 1. Ввести логин и пароль 2. Нажать кнопку «Войти» 3. Убедиться, что открылась главная форма роли |
| **Тестовые данные** | * Логин: 111 * Пароль: 111 |
| **Ожидаемый результат** | Форма авторизации закрывается, открывается главная форма соответствующей роли |
| **Фактический результат** | Форма авторизации закрывается, открывается главная форма соответствующей роли |
| **Предварительное условие** | - |
| **Постусловие** | Открыта главная форма соответствующей роли |
| **Статус** | Зачет |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | ATM |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Нестерович Николай |
| **Дата(ы) теста** | 01.02.23 |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет**  **Тестирования** | Низкий |
| **Заголовок/название теста** | Проверка авторизации с неверно введенными логином и паролем |
| **Краткое изложение теста** | На форме авторизации отображается текст «Неверный логин или пароль» |
| **Этапы теста** | 1. Ввести логин и пароль 2. Нажать кнопку «Войти» 3. Убедиться, что отобразился текст ошибки |
| **Тестовые данные** | * Логин: dad2wd * Пароль: 2daw2wad2ad |
| **Ожидаемый результат** | На форме авторизации отображается текст «Неверный логин или пароль» |
| **Фактический результат** | На форме авторизации отображается текст «Неверный логин или пароль» |
| **Предварительное условие** | - |
| **Постусловие** | Форма авторизации остается открыта |
| **Статус** | Зачет |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | ATM |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Нестерович Николай |
| **Дата(ы) теста** | 01.02.23 |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет**  **Тестирования** | Низкий |
| **Заголовок/название теста** | Проверка авторизации с не введенными логином и паролем |
| **Краткое изложение теста** | На форме авторизации отображается текст «Введите логин и пароль» |
| **Этапы теста** | 1. Ввести логин и пароль 2. Нажать кнопку «Войти» 3. Убедиться, что отобразился текст ошибки |
| **Тестовые данные** | * Логин: * Пароль: |
| **Ожидаемый результат** | На форме авторизации отображается текст «Введите логин и пароль» |
| **Фактический результат** | На форме авторизации отображается текст «Введите логин и пароль» |
| **Предварительное условие** | - |
| **Постусловие** | Форма авторизации остается открыта |
| **Статус** | Зачет |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | ATM |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Нестерович Николай |
| **Дата(ы) теста** | 01.02.23 |
| **Тестовый пример #** | TC\_UI\_4 |
| **Приоритет**  **Тестирования** | Низкий |
| **Заголовок/название теста** | Проверка добавления клиента |
| **Краткое изложение теста** | Менеджер должен иметь возможность добавить клиента |
| **Этапы теста** | 1. Авторизоваться под ролью «Менеджер» 2. Нажать кнопку «Добавить клиента» 3. Ввести все данные клиента 4. Нажать кнопку «Добавить клиента» 5. Убедиться, что новый клиент отображается в таблице |
| **Тестовые данные** | * ФИО: тест * Номер карты: 1234123412341234 * Адрес: тест * Банк: [первый в списке] |
| **Ожидаемый результат** | Форма «Добавление клиента» закрылась, новый клиент отображается в таблице |
| **Фактический результат** | Форма «Добавление клиента» закрылась, новый клиент отображается в таблице |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Статус** | Зачет |
| **Примечания/комментарии** | - |
| **Поле** | **Описание** |
| **Название проекта** | ATM |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Нестерович Николай |
| **Дата(ы) теста** | 01.02.23 |
| **Тестовый пример #** | TC\_UI\_5 |
| **Приоритет**  **Тестирования** | Низкий |
| **Заголовок/название теста** | Проверка удаления клиента |
| **Краткое изложение теста** | Менеджер должен иметь возможность удалить клиента |
| **Этапы теста** | 1. Авторизоваться под ролью «Менеджер» 2. Выбрать клиента 3. Нажать кнопку «Удалить» 4. В появившемся окне подтверждения нажать «ОК» 5. Убедиться, что клиент больше не отображается в таблице |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Клиент больше не отображается в таблице |
| **Фактический результат** | Клиент больше не отображается в таблице |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Статус** | Зачет |
| **Примечания/комментарии** | - |

Приложение В

**Текст программы**

**Модуль accountant.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class accountant : Form

{

public accountant()

{

InitializeComponent();

userNameLB.Text = $"login: {db.userInfo[1]}";

string[] role = db.Select($"SELECT \* FROM Роли WHERE ID = {db.userInfo[3]}")[0];

int roleId = Convert.ToInt32(role[0]);

userRoleLB.Text = role[1];

}

private void transactionsBTN\_Click(object sender, EventArgs e)

{

transactions newForm = new transactions();

newForm.Show();

}

private void reportAtmsBTN\_Click(object sender, EventArgs e)

{

reportAtms newForm = new reportAtms();

newForm.Show();

}

private void reportFeesBTN\_Click(object sender, EventArgs e)

{

reportFees newForm = new reportFees();

newForm.Show();

}

private void accountant\_FormClosed(object sender, FormClosedEventArgs e)

{

login newForm = new login();

newForm.Show();

}

}

}

**Модуль addAtm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class addAtm : Form

{

List<string[]> banks;

string[] atmInfo;

public addAtm(int atmID = -1)

{

InitializeComponent();

banks = db.Select($"SELECT \* FROM Банки " +

$"WHERE is\_archived != 1");

WSLib.utils.UpdateCB(banksCB, banks, 1);

atmInfo = new string[] { };

if (atmID != -1)

{

atmInfo = db.Select($"SELECT Банкоматы.ID, Банкоматы.Адрес, Банки.Название " +

$"FROM Банкоматы " +

$"INNER JOIN Банки ON Банкоматы.ID\_банка = Банки.ID " +

$"WHERE Банкоматы.ID = {atmID}")[0];

Text = "Изменение банкомата";

addAtmBTN.Text = "Изменить банкомат";

addressTB.Text = atmInfo[1];

banksCB.Text = atmInfo[2];

}

}

private void addAtmBTN\_Click(object sender, EventArgs e)

{

if (addressTB.Text == "" || banksCB.Text == "")

{

MessageBox.Show(

"Не все данные банкомата введены!",

"Предупреждение",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else if (atmInfo.Length == 0)

{

db.Select($"INSERT INTO Банкоматы (Адрес, ID\_банка, is\_archived) " +

$"VALUES ('{addressTB.Text}', {banks[banksCB.SelectedIndex][0]}, 0)");

this.Hide();

atms newForm = new atms();

newForm.Show();

}

else

{

db.Select($"UPDATE Банкоматы SET Адрес = '{addressTB.Text}', ID\_банка = {banks[banksCB.SelectedIndex][0]} " +

$"WHERE ID = {atmInfo[0]}");

this.Hide();

atms newForm = new atms();

newForm.Show();

}

}

private void addAtm\_FormClosed(object sender, FormClosedEventArgs e)

{

atms newForm = new atms();

newForm.Show();

}

}

}

**Модуль addBank.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class addBank : Form

{

string[] bankInfo;

public addBank(int bankID = -1)

{

InitializeComponent();

bankInfo = new string[] { };

if (bankID != -1)

{

bankInfo = db.Select($"SELECT \* FROM Банки WHERE ID = {bankID}")[0];

Text = "Изменение банка";

addBankBTN.Text = "Изменить банк";

nameTB.Text = bankInfo[1];

addressTB.Text = bankInfo[2];

}

}

private void addBankBTN\_Click(object sender, EventArgs e)

{

if (nameTB.Text == "" || addressTB.Text == "")

{

MessageBox.Show(

"Не все данные банка введены!",

"Предупреждение",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else if (bankInfo.Length == 0)

{

db.Select($"INSERT INTO Банки (Название, Юридический\_адрес, is\_archived) " +

$"VALUES ('{nameTB.Text}', '{addressTB.Text}', 0)");

this.Hide();

banks newForm = new banks();

newForm.Show();

}

else

{

db.Select($"UPDATE Банки SET Название = '{nameTB.Text}', Юридический\_адрес = '{addressTB.Text}' " +

$"WHERE ID = {bankInfo[0]}");

this.Hide();

banks newForm = new banks();

newForm.Show();

}

}

private void addBank\_FormClosed(object sender, FormClosedEventArgs e)

{

banks newForm = new banks();

newForm.Show();

}

}

}

**Модуль addClient.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class addClient : Form

{

List<string[]> banks;

string[] clientInfo;

public addClient(int clientID = -1)

{

InitializeComponent();

banks = db.Select("SELECT \* FROM Банки " +

$"WHERE is\_archived != 1");

WSLib.utils.UpdateCB(banksCB, banks, 1);

clientInfo = new string[] { };

if (clientID != -1)

{

clientInfo = db.Select($"SELECT Клиенты.ID, ФИО, Номер\_карты, Адрес, Банки.Название AS Банк " +

$"FROM Клиенты INNER JOIN Банки ON Клиенты.ID\_банка = Банки.ID " +

$"WHERE Клиенты.ID = {clientID}")[0];

Text = "Изменение клиента";

addClientBTN.Text = "Изменить клиента";

nameTB.Text = clientInfo[1];

cardNUD.Value = Convert.ToInt64(clientInfo[2]);

addressTB.Text = clientInfo[3];

banksCB.Text = clientInfo[4];

}

}

private void addClientBTN\_Click(object sender, EventArgs e)

{

if (nameTB.Text == "" || cardNUD.Value == 1000000000000000 || addressTB.Text == "" || banksCB.Text == "")

{

MessageBox.Show(

"Не все данные клиента введены!",

"Предупреждение",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else if (clientInfo.Length == 0)

{

db.Select($"INSERT INTO Клиенты (ФИО, Номер\_карты, Адрес, ID\_банка, is\_archived) " +

$"VALUES ('{nameTB.Text}', {cardNUD.Value}, '{addressTB.Text}', {banks[banksCB.SelectedIndex][0]}, 0)");

this.Hide();

manager newForm = new manager();

newForm.Show();

}

else

{

db.Select($"UPDATE Клиенты SET ФИО = '{nameTB.Text}', Номер\_карты = {cardNUD.Value}, Адрес = '{addressTB.Text}', ID\_банка = {banks[banksCB.SelectedIndex][0]} " +

$"WHERE ID = {clientInfo[0]}");

this.Hide();

manager newForm = new manager();

newForm.Show();

}

}

private void addClient\_FormClosed(object sender, FormClosedEventArgs e)

{

manager newForm = new manager();

newForm.Show();

}

}

}

**Модуль administrator.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class administrator : Form

{

public administrator()

{

InitializeComponent();

userNameLB.Text = $"login: {db.userInfo[1]}";

string[] role = db.Select($"SELECT \* FROM Роли WHERE ID = {db.userInfo[3]}")[0];

int roleId = Convert.ToInt32(role[0]);

userRoleLB.Text = role[1];

}

private void banksBTN\_Click(object sender, EventArgs e)

{

banks newForm = new banks();

newForm.Show();

this.Hide();

}

private void atmsBTN\_Click(object sender, EventArgs e)

{

atms newForm = new atms();

newForm.Show();

this.Hide();

}

private void administrator\_FormClosed(object sender, FormClosedEventArgs e)

{

login newForm = new login();

newForm.Show();

this.Hide();

}

}

}

**Модуль atms.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class atms : Form

{

List<string[]> atmsList;

public atms()

{

InitializeComponent();

UpdateAll();

}

private void UpdateAll()

{

atmsList = db.Select($"SELECT Банкоматы.ID, Банкоматы.Адрес, Банки.Название " +

$"FROM Банкоматы INNER JOIN Банки ON Банкоматы.ID\_банка = Банки.ID " +

$"WHERE Банкоматы.is\_archived != 1");

WSLib.utils.UpdateTable(dataGridView1, atmsList, new string[] { "ID", "Адрес", "Банк" });

}

private void addAtmBTN\_Click(object sender, EventArgs e)

{

addAtm newForm = new addAtm();

newForm.Show();

this.Hide();

}

private void updateAtmBTN\_Click(object sender, EventArgs e)

{

addAtm newForm = new addAtm(Convert.ToInt32(atmsList[dataGridView1.CurrentCell.RowIndex][0]));

newForm.Show();

this.Hide();

}

private void deleteAtmBTN\_Click(object sender, EventArgs e)

{

if (MessageBox.Show(

$"Удалить банкомат: {atmsList[dataGridView1.CurrentCell.RowIndex][1]}?",

"Подтверждение",

MessageBoxButtons.OKCancel,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.OK)

{

MessageBox.Show(

$"Банкомат {atmsList[dataGridView1.CurrentCell.RowIndex][1]} удален!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

db.Select($"UPDATE Банкоматы SET is\_archived = 1 " +

$"WHERE ID = {atmsList[dataGridView1.CurrentCell.RowIndex][0]}");

UpdateAll();

}

}

private void atms\_FormClosed(object sender, FormClosedEventArgs e)

{

administrator newForm = new administrator();

newForm.Show();

}

}

}

**Модуль banks.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class banks : Form

{

List<string[]> banksList;

public banks()

{

InitializeComponent();

UpdateAll();

}

private void UpdateAll()

{

banksList = db.Select($"SELECT ID, Название, Юридический\_адрес FROM Банки " +

$"WHERE Банки.is\_archived != 1");

WSLib.utils.UpdateTable(dataGridView1, banksList, new string[] { "ID", "Название", "Юридический адрес" });

}

private void addBankBTN\_Click(object sender, EventArgs e)

{

addBank newForm = new addBank();

newForm.Show();

this.Hide();

}

private void updateBankBTN\_Click(object sender, EventArgs e)

{

addBank newForm = new addBank(Convert.ToInt32(banksList[dataGridView1.CurrentCell.RowIndex][0]));

newForm.Show();

this.Hide();

}

private void deleteBankBTN\_Click(object sender, EventArgs e)

{

if (MessageBox.Show(

$"Удалить банк: {banksList[dataGridView1.CurrentCell.RowIndex][1]}?",

"Подтверждение",

MessageBoxButtons.OKCancel,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.OK)

{

MessageBox.Show(

$"Банк {banksList[dataGridView1.CurrentCell.RowIndex][1]} удален!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

db.Select($"UPDATE Банки SET is\_archived = 1 " +

$"WHERE ID = {banksList[dataGridView1.CurrentCell.RowIndex][0]}");

UpdateAll();

}

}

private void banks\_FormClosed(object sender, FormClosedEventArgs e)

{

administrator newForm = new administrator();

newForm.Show();

}

}

}

**Модуль login.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class login : Form

{

List<string[]> userInfo;

public login()

{

InitializeComponent();

passwordTB.UseSystemPasswordChar = !passwordTB.UseSystemPasswordChar;

}

private void loginBTN\_Click(object sender, EventArgs e)

{

if (loginTB.Text.Length == 0 || passwordTB.Text.Length == 0)

{

errorLB.Text = "Введите логин и пароль";

}

else

{

userInfo = db.Select($"SELECT \* FROM Пользователи WHERE Логин = '{loginTB.Text}' AND Пароль = '{passwordTB.Text}'");

if (userInfo.Count == 0)

{

errorLB.Text = "Неверный логин или пароль";

}

else

{

db.userInfo = userInfo[0];

if (db.userInfo[3] == "1") // Бухгалтер - Отчеты

{

accountant newForm = new accountant();

newForm.Show();

this.Hide();

}

if (db.userInfo[3] == "2") // Менеджер - Клиенты

{

manager newForm = new manager();

newForm.Show();

this.Hide();

}

if (db.userInfo[3] == "3") // Администратор - Банки и банкоматы

{

administrator newForm = new administrator();

newForm.Show();

this.Hide();

}

}

}

}

private void passwordTB\_change\_visibility(object sender, EventArgs e)

{

passwordTB.UseSystemPasswordChar = !passwordTB.UseSystemPasswordChar;

}

private void login\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

**Модуль manager.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class manager : Form

{

List<string[]> clients;

public manager()

{

InitializeComponent();

userNameLB.Text = $"login: {db.userInfo[1]}";

string[] role = db.Select($"SELECT \* FROM Роли WHERE ID = {db.userInfo[3]}")[0];

int roleId = Convert.ToInt32(role[0]);

userRoleLB.Text = role[1];

UpdateAll();

}

private void UpdateAll()

{

clients = db.Select($"SELECT Клиенты.ID, ФИО, Номер\_карты, Адрес, Банки.Название AS Банк " +

$"FROM Клиенты INNER JOIN Банки ON Клиенты.ID\_банка = Банки.ID " +

$"WHERE Клиенты.is\_archived != 1");

WSLib.utils.UpdateTable(dataGridView1, clients, new string[] { "ID", "ФИО", "Номер карты", "Адрес", "Банк" });

}

private void addClientBTN\_Click(object sender, EventArgs e)

{

addClient newForm = new addClient();

newForm.Show();

this.Hide();

}

private void updateClientBTN\_Click(object sender, EventArgs e)

{

addClient newForm = new addClient(Convert.ToInt32(clients[dataGridView1.CurrentCell.RowIndex][0]));

newForm.Show();

this.Hide();

}

private void deleteClientBTN\_Click(object sender, EventArgs e)

{

if (MessageBox.Show(

$"Удалить клиента: {clients[dataGridView1.CurrentCell.RowIndex][1]}?",

"Подтверждение",

MessageBoxButtons.OKCancel,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.OK)

{

MessageBox.Show(

$"Клиент {clients[dataGridView1.CurrentCell.RowIndex][1]} удален!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

db.Select($"UPDATE Клиенты SET is\_archived = 1 " +

$"WHERE ID = {clients[dataGridView1.CurrentCell.RowIndex][0]}");

UpdateAll();

}

}

private void manager\_FormClosed(object sender, FormClosedEventArgs e)

{

login newForm = new login();

newForm.Show();

this.Hide();

}

}

}

**Модуль reportAtms.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class reportAtms : Form

{

List<string[]> atmsData;

List<string[]> banks;

public reportAtms()

{

InitializeComponent();

banks = db.Select("SELECT \* FROM Банки " +

$"WHERE is\_archived != 1");

banksCB.Items.Add("Все банки");

for (var i = 0; i < banks.Count; i++)

{

banksCB.Items.Add(banks[i][1]);

}

banksCB.SelectedIndex = 0;

UpdateAll();

}

private void UpdateAll()

{

string request = $"SELECT Банкоматы.Адрес, COUNT(Операции.ID), SUM(Операции.Сумма) " +

$"FROM Операции INNER JOIN Клиенты ON Операции.ID\_клиента = Клиенты.ID " +

$"INNER JOIN Банкоматы ON Операции.ID\_банкомата = Банкоматы.ID " +

$"INNER JOIN Банки ON Банкоматы.ID\_банка = Банки.ID ";

if (banksCB.SelectedIndex != 0)

{

request += $"WHERE Банки.ID = {banks[banksCB.SelectedIndex - 1][0]} " +

$"AND Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' ";

}

else

{

request += $"WHERE Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' ";

}

request += "AND Банкоматы.is\_archived != 1";

request += $"GROUP BY Банкоматы.Адрес ";

atmsData = db.Select(request);

WSLib.utils.UpdateTable(dataGridView1, atmsData, new string[] { "Банкомат", "Количество операций", "Сумма" });

WSLib.utils.GenerateDiagram(chart1, dataGridView1, 0, 1, "Количество операций");

WSLib.utils.GenerateDiagram(chart2, dataGridView1, 0, 2, "Сумма");

}

private void updateDGVs(object sender, EventArgs e)

{

UpdateAll();

}

private void saveReportPDF\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

DateTime now = DateTime.Now;

string fileName = $"report\_atms\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

string[] pdfTablesNames = { "Статистика по банкоматам" };

string[][] pdfTablesColumns = { new string[] { "Банкомат", "Количество операций", "Сумма" } };

List<string[]>[] pdfData = {

new List<string[]> { new string[] { $"{now.ToString("yyyy-MM-dd")}" } },

new List<string[]> { new string[] { $"Банк: {pdfBank}" } },

atmsData,

};

WSLib.utils.pdfWriter(fileName, pdfTablesNames, pdfData, pdfTablesColumns);

}

private void saveReportCSV\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

DateTime now = DateTime.Now;

string fileName = $"report\_atms\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

WSLib.utils.csvWriter(fileName, atmsData, new string[] { "Банкомат", "Количество операций", "Сумма" });

MessageBox.Show(

$"Отчет {fileName} сохранен!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

**Модуль reportFees.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class reportFees : Form

{

List<string[]> feesData;

List<string[]> banks;

public reportFees()

{

InitializeComponent();

banks = db.Select("SELECT \* FROM Банки " +

$"WHERE is\_archived != 1");

banksCB.Items.Add("Все банки");

for (var i = 0; i < banks.Count; i++)

{

banksCB.Items.Add(banks[i][1]);

}

banksCB.SelectedIndex = 0;

UpdateAll();

}

private void UpdateAll()

{

string request = $"SELECT Операции.ID, Клиенты.ФИО, Банкоматы.Адрес, Операции.Дата\_время, Операции.Сумма " +

$"FROM Операции INNER JOIN Клиенты ON Операции.ID\_клиента = Клиенты.ID " +

$"INNER JOIN Банкоматы ON Операции.ID\_банкомата = Банкоматы.ID " +

$"INNER JOIN Банки ON Банкоматы.ID\_банка = Банки.ID " +

$"WHERE Банки.ID == Клиенты.ID\_банка ";

if (banksCB.SelectedIndex != 0)

{

request += $"AND Банки.ID = {banks[banksCB.SelectedIndex - 1][0]} " +

$"AND Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}'";

}

else

{

request += $"AND Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' ";

}

request = request.Replace("==", "!=");

if (feeCB.Checked == false)

{

request = request.Replace("!=", "=");

}

feesData = db.Select(request);

WSLib.utils.UpdateTable(dataGridView1, feesData, new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" });

}

private void updateDGVs(object sender, EventArgs e)

{

UpdateAll();

}

private void saveReportPDF\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

string pdfFee = "Да";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

if (feeCB.Checked == false) pdfFee = "Нет";

DateTime now = DateTime.Now;

string fileName = $"report\_fees\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

string[] pdfTablesNames = { "Статистика по банкоматам" };

string[][] pdfTablesColumns = { new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" } };

List<string[]>[] pdfData = {

new List<string[]> { new string[] { $"{now.ToString("yyyy-MM-dd")}" } },

new List<string[]> { new string[] { $"Банк: {pdfBank}" } },

new List<string[]> { new string[] { $"Комиссия: {pdfFee}" } },

feesData,

};

WSLib.utils.pdfWriter(fileName, pdfTablesNames, pdfData, pdfTablesColumns);

}

private void saveReportCSV\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

DateTime now = DateTime.Now;

string fileName = $"report\_fees\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

WSLib.utils.csvWriter(fileName, feesData, new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" });

MessageBox.Show(

$"Отчет {fileName} сохранен!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

**Модуль transactions.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ATM

{

public partial class transactions : Form

{

List<string[]> transactionsList;

List<string[]> diagramData;

List<string[]> banks;

public transactions()

{

InitializeComponent();

banks = db.Select("SELECT \* FROM Банки " +

$"WHERE is\_archived != 1");

banksCB.Items.Add("Все банки");

for (var i = 0; i < banks.Count; i++)

{

banksCB.Items.Add(banks[i][1]);

}

banksCB.SelectedIndex = 0;

UpdateAll();

}

private void UpdateAll()

{

string request = $"SELECT Операции.ID, Клиенты.ФИО, Банкоматы.Адрес, Операции.Дата\_время, Операции.Сумма " +

$"FROM Операции INNER JOIN Клиенты ON Операции.ID\_клиента = Клиенты.ID " +

$"INNER JOIN Банкоматы ON Операции.ID\_банкомата = Банкоматы.ID ";

string diagramRequest = $"SELECT COUNT(Операции.ID), Операции.Дата\_время " +

$"FROM Операции INNER JOIN Клиенты ON Операции.ID\_клиента = Клиенты.ID " +

$"INNER JOIN Банкоматы ON Операции.ID\_банкомата = Банкоматы.ID ";

if (banksCB.SelectedIndex != 0)

{

request += $"INNER JOIN Банки ON Клиенты.ID\_банка = Банки.ID " +

$"WHERE Банки.ID = {banks[banksCB.SelectedIndex - 1][0]} " +

$"AND Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}'";

diagramRequest += $"INNER JOIN Банки ON Клиенты.ID\_банка = Банки.ID " +

$"WHERE Банки.ID = {banks[banksCB.SelectedIndex - 1][0]} " +

$"AND Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}'";

}

else

{

request += $"WHERE Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' ";

diagramRequest += $"WHERE Дата\_время BETWEEN '{fromDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' AND '{toDTP.Value.ToString("yyyy-MM-dd hh:mm:ss")}' ";

}

transactionsList = db.Select(request);

WSLib.utils.UpdateTable(dataGridView1, transactionsList, new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" });

diagramRequest += $"GROUP BY Операции.Дата\_время";

diagramData = db.Select(diagramRequest);

WSLib.utils.UpdateTable(dataGridView2, diagramData, new string[] { "Количество операций", "Дата" });

WSLib.utils.GenerateDiagram(chart1, dataGridView2, 1, 0, "Количество операций");

}

private void updateDGVs(object sender, EventArgs e) { UpdateAll(); }

private void saveReportPDF\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

DateTime now = DateTime.Now;

string fileName = $"report\_transactions\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

string[] pdfTablesNames = { "Статистика по банкоматам" };

string[][] pdfTablesColumns = { new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" } };

List<string[]>[] pdfData = {

new List<string[]> { new string[] { $"{now.ToString("yyyy-MM-dd")}" } },

new List<string[]> { new string[] { $"Банк: {pdfBank}" } },

transactionsList,

};

WSLib.utils.pdfWriter(fileName, pdfTablesNames, pdfData, pdfTablesColumns);

}

private void saveReportCSV\_Click(object sender, EventArgs e)

{

string pdfBank = "Все банки";

if (banksCB.SelectedIndex != 0) pdfBank = banks[banksCB.SelectedIndex - 1][0];

DateTime now = DateTime.Now;

string fileName = $"report\_transactions\_\_{now.ToString("MM\_dd\_\_hh\_mm\_ss")}";

WSLib.utils.csvWriter(fileName, transactionsList, new string[] { "ID", "Клиент", "Адрес банкомата", "Дата", "Сумма" });

MessageBox.Show(

$"Отчет {fileName} сохранен!",

"Подтверждение",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

**Модуль db.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

namespace ATM

{

class db

{

public static string[] userInfo = new string[] { "-1", "test" };

public static SqlConnection connection = new SqlConnection(@"Data Source=DESKTOP-CCE6AH1\SQLEXPRESS; Initial Catalog=ATM;Integrated Security=true");

// Запрос к БД и возврат результирующей таблицы

public static List<string[]> Select(string request)

{

connection.Open();

var command = new SqlCommand(String.Format(request), connection);

var tempList = new List<string>(); // Запись из БД

var result = new List<string[]>(); // Результат запроса к БД

using (var reader = command.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

tempList.Clear();

for (var i = 0; i < reader.FieldCount; i++)

{

tempList.Add(Convert.ToString(reader.GetValue(i)));

}

var tempArray = new string[tempList.Count];

tempList.CopyTo(tempArray);

result.Add(tempArray);

}

}

}

connection.Close();

return result;

}

}

}

**Текст библиотеки классов**

**Модуль utils.cs**

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

using iTextSharp.text;

using System.IO;

using iTextSharp.text.pdf;

using System.Windows.Forms.DataVisualization.Charting;

namespace WSLib

{

public class utils

{

/// <summary>

/// Метод генерации диаграмм.

/// </summary>

/// <param name="chart">Ссылка на заполняемый объект Chart</param>

/// <param name="table">Ссылка на объект DataGridView с исходными данными</param>

/// <param name="xIndex">Индекс поля таблицы для оси X</param>

/// <param name="yIndex">Индекс поля таблицы для оси Y</param>

/// <param name="legend">Название элемента легенды</param>

/// <returns></returns>

public static void GenerateDiagram(Chart chart, DataGridView table, int xIndex, int yIndex, string legend)

{

chart.Series.Clear();

chart.Series.Add(new Series(legend) { ChartType = SeriesChartType.Column });

string[] xValues = new string[table.Rows.Count];

double[] yValues = new double[table.Rows.Count];

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

{

xValues[i] = table[xIndex, i].Value.ToString();

yValues[i] = Convert.ToDouble(table[yIndex, i].Value);

}

chart.Series[legend].Points.DataBindXY(xValues, yValues);

}

/// <summary>

/// Метод обновления элементов выпадающего списка.

/// </summary>

/// <param name="box">Ссылка на заполняемый объект ComboBox</param>

/// <param name="data">Исходный набор данных</param>

/// <param name="index">Индекс поля таблицы</param>

/// <returns></returns>

public static int UpdateCB(ComboBox box, List<string[]> data, int index)

{

if (data.Count == 0) return 0;

box.Items.Clear();

for (var i = 0; i < data.Count; i++)

{

box.Items.Add(data[i][index]);

}

return 1;

}

/// <summary>

/// Метод обновления данных в таблице.

/// </summary>

/// <param name="table">Ссылка на заполняемый объект DataGridView</param>

/// <param name="data">Исходный набор данных</param>

/// <param name="columns">Список заголовков столбцов таблицы</param>

/// <returns></returns>

public static void UpdateTable(DataGridView table, List<string[]> data, string[] columns)

{

if (data.Count > 0)

{

string[] columnNames = columns;

table.ColumnCount = data[0].Length;

table.RowCount = data.Count;

for (var i = 0; i < columnNames.Length; i++) { table.Columns[i].Name = columnNames[i]; }

for (var i = 0; i < data.Count; i++)

{

for (var j = 0; j < data[0].Length; j++)

{

table.Rows[i].Cells[j].Value = data[i][j];

}

}

}

else { table.Rows.Clear(); }

}

/// <summary>

/// Метод записи данных в PDF.

/// </summary>

/// <param name="file">Название будущего файла</param>

/// <param name="tableNames">Список заголовков таблицы</param>

/// <param name="data">Исходный набор данных</param>

/// <param name="columns">Список заголовков столбцов таблицы</param>

/// <returns></returns>

public static void pdfWriter(string file, string[] tableNames, List<string[]>[] data, string[][] columns)

{

Document pdf = new Document();

PdfWriter.GetInstance(pdf, new FileStream($"..\\..\\..\\Docs PDF\\{file}.pdf", FileMode.Create));

pdf.Open();

BaseFont baseFont = BaseFont.CreateFont(@"C:\Windows\Fonts\arial.ttf", BaseFont.IDENTITY\_H, BaseFont.NOT\_EMBEDDED);

iTextSharp.text.Font font = new iTextSharp.text.Font(baseFont, iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);

int tablesCount = 0;

for (int textTable = 0; textTable < data.Length; textTable++)

{

if (data[textTable].Count == 1 && data[textTable][0].Length == 1)

{

pdf.Add(new Paragraph(data[textTable][0][0], font));

}

else

{

PdfPTable table = new PdfPTable(data[textTable][0].Length);

PdfPCell cell = new PdfPCell(new Phrase(tableNames[tablesCount], font));

cell.Colspan = data[textTable][0].Length;

cell.HorizontalAlignment = 1;

cell.Border = 0;

table.AddCell(cell);

for (int i = 0; i < data[textTable][0].Length; i++)

{

cell = new PdfPCell(new Phrase(columns[tablesCount][i], font));

cell.BackgroundColor = BaseColor.YELLOW;

table.AddCell(cell);

}

for (int i = 0; i < data[textTable].Count; i++)

{

for (int j = 0; j < data[textTable][i].Length; j++)

{

cell = new PdfPCell(new Phrase(data[textTable][i][j], font));

table.AddCell(cell);

}

}

pdf.Add(table);

tablesCount++;

}

}

pdf.Close();

System.Diagnostics.Process.Start($"..\\..\\..\\Docs CSV\\{file}.pdf");

}

/// <summary>

/// Метод записи данных в CSV.

/// </summary>

/// <param name="file">Название будущего файла</param>

/// <param name="data">Исходный набор данных</param>

/// <param name="columns">Список заголовков столбцов таблицы</param>

/// <returns></returns>

public static void csvWriter(string file, List<string[]> data, string[] columns)

{

string \_data = "";

for (int i = 0; i < data[0].Length; i++)

{

\_data += $"{columns[i]};";

}

\_data += "\t\n";

for (int i = 0; i < data.Count; i++)

{

for (int j = 0; j < data[i].Length; j++)

{

\_data += $"{data[i][j]};";

}

\_data += "\t\n";

}

StreamWriter sw = new StreamWriter($"..\\..\\..\\Docs CSV\\{file}.csv", false, Encoding.GetEncoding("windows-1251"));

sw.Write(\_data);

sw.Close();

}

}

}

Приложение Г

CREATE TABLE Роли (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

Название nvarchar(50),

is\_archived bit,

)

CREATE TABLE Пользователи (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

Логин nvarchar(30),

Пароль nvarchar(30),

ID\_роли int NOT NULL,

is\_archived bit,

FOREIGN KEY (ID\_роли) REFERENCES Роли (ID),

)

CREATE TABLE Банки (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

Название nvarchar(50),

Юридический\_адрес nvarchar(150),

is\_archived bit,

)

CREATE TABLE Клиенты (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

ФИО nvarchar(50),

Номер\_карты bigint,

Адрес nvarchar(150),

ID\_банка int NOT NULL,

is\_archived bit,

FOREIGN KEY (ID\_банка) REFERENCES Банки (ID),

)

CREATE TABLE Банкоматы (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

Адрес nvarchar(150),

ID\_банка int NOT NULL,

is\_archived bit,

FOREIGN KEY (ID\_банка) REFERENCES Банки (ID),

)

CREATE TABLE Операции (

ID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

ID\_клиента int NOT NULL,

ID\_банкомата int NOT NULL,

Дата\_время date,

Сумма money,

is\_archived bit,

FOREIGN KEY (ID\_клиента) REFERENCES Клиенты (ID),

FOREIGN KEY (ID\_банкомата) REFERENCES Банкоматы (ID),

)

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/6 группы

Нестерович Николай Дмитриевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «09» января 2023 г. по «04» февраля 2023 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| Участие в выработке требований к программному обеспечению | 18 |  |
| Стадии проектирования программного обеспечения | 42 |  |
| Разработка модулей программного обеспечения | 48 |  |
| Тестирование программных модулей и их интеграции | 18 |  |
| Разработка программной документации и стандарты кодирования | 18 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Челищева Л.Д. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «04» февраля 2023 г.