**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 42919/2 группы

Филимонов Владислав Анатольевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

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

Санкт-Петербург

2024

**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 42919/2 группы

Филимонов Владислав Анатольевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

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

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 21**

Задание выдал «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Филимонов В.А.(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 42919/2 группы

Филимонов Владислав Анатольевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 23.09.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 24.09.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 25.09.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 26.09.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 27.09.2024 | Проектирование интерфейса пользователя. |  |
| 28.09.2024 | Разработка схемы алгоритма программного продукта. |  |
| 30.09.2024 | Разработка модели данных. |  |
| 01.10.2024 | Разработка программного продукта. Создание функциональности ПО. |  |
| 02.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 03.10.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 04.10.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 05.10.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

Оглавление

[ВВЕДЕНИЕ 6](#_Toc178926044)

[Задание 1. Техническое задание 7](#_Toc178926045)

[Задание 2. Разработка алгоритмов и диаграмм 10](#_Toc178926046)

[Задание 3. Макеты 21](#_Toc178926047)

[Задание 4. Разработка 25](#_Toc178926048)

[Задание 5. Тестирование 31](#_Toc178926049)

[Задание 6. GitHub. Оценка проекта 35](#_Toc178926050)

[ЗАКЛЮЧЕНИЕ 36](#_Toc178926051)

[ПРИЛОЖЕНИЕ А 37](#_Toc178926052)

[ПРИЛОЖЕНИЕ Б 52](#_Toc178926053)

ВВЕДЕНИЕ

На данной практике я планирую продолжить разработку и тестирование приложения для управления клиентами, а также улучшить его функциональность и производительность, чтобы оно могло быть использовано в реальных условиях, и получить опыт в разработке и тестировании приложений, а также улучшить свои навыки программирования и анализа данных.

Задание 1. Техническое задание

Описание предметной области

Основной целью данного модуля является автоматизация процесса учета и обработки залоговых операций в ломбарде. Модуль должен обеспечить прозрачное и эффективное управление залоговыми предметами, отслеживание текущего статуса залогов и оптимизацию работы сотрудников ломбарда.

Разработка программного модуля для учета залоговых операций в ломбарде — это процесс создания программного продукта, который позволяет ломбардам эффективно отслеживать и управлять залоговыми предметами, включая учет операций по залогу, выдачу и возврат займов, начисление процентов, реализацию невостребованного имущества и ведение бухгалтерского, налогового и управленческого учета в соответствии с отраслевыми стандартами и законодательными требованиями.

Задача разработки такого программного модуля для ломбарда состоит в создании удобного и эффективного инструмента для учета, отслеживания и управления залоговыми операциями, что помогает минимизировать время обработки операций, обеспечивает бесперебойную работу в ломбарде и повышает удовлетворенность клиентов.

Этот модуль должен обеспечить прозрачное управление залоговыми предметами, автоматизировать процесс регистрации, хранения и погашения залогов, а также предоставлять полную отчетность, анализ процесса учета залоговых операций и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1. Залоговая операция
   * Информация, предоставленная клиентом о залоговом предмете, включая тип предмета, описание, оценочную стоимость, личную информацию клиента (ФИО, номер телефона).
2. Регистрация залоговой операции
   * Приём и регистрация залоговой операции сотрудником ломбарда в системе учета.
   * Присвоение уникального идентификатора операции, сохранение информации о залоговом предмете.
3. Обработка залоговой операции
   * Анализ залоговой операции, определение условий залога и назначение сотрудника для обработки.
4. Хранение и мониторинг залоговых предметов
   * Фактическое хранение залоговых предметов и мониторинг их состояния.
5. Погашение залога
   * Процесс погашения залога, включая возврат залогового предмета клиенту после погашения кредита.
   * Фиксация информации о погашении залога
6. Отчётность и информирование
   * Фиксация и отчёт о выполненных операциях, включая информацию о залоговых предметах и сроках хранения.
7. Мониторинг и анализ
   * Контроль и анализ процесса учета залоговых операций.
   * Отслеживание качества обслуживания, расходов и других параметров для оптимизации и улучшения процесса.

**Техническое задание**

1. Общие сведения
   1. Наименование проекта: Разработка программного модуля для учета заявок на сдачу предметов в ломбард.
   2. Заказчик: Ломбард «ЛомбУниверсал».
   3. Исполнитель: Сервисный центр «МяуГрад».
2. Функциональные требования
   1. Возможность добавления залоговых операций в базу данных:
      * Номер операции;
      * Дата добавления;
      * Тип залогового предмета;
      * Описание предмета;
      * Оценочная стоимость;
      * ФИО клиента;
      * Номер телефона;
      * Статус операции (новая операция, в процессе, погашена).
   2. Возможность редактирования залоговых операций:
      * Изменение статуса операции (погашена, продлена, закрыта);
      * Изменение описания предмета;
      * Изменение ответственного за обработку операции.
   3. Возможность отслеживания статуса операций:
      * Отображение списка операций;
      * Поиск операции по номеру или по параметрам.
   4. Возможность назначения ответственных за обработку операций:
      1. Добавление сотрудника к операции;
      2. Отслеживание состояния работы;
   5. Расчет статистики работы ломбарда:
      * Количество обработанных операций;
      * Среднее время обработки операций;
      * Статистика по типам залоговых предметов.
3. Нефункциональные требования
   1. Кроссплатформенность:
      * -Поддержка работы на ОС семейства Windows.
   2. Безопасность:
      * Логин и пароль для доступа к приложению;
      * Доступ к данным должен быть ограничен в зависимости от роли пользователя.
   3. Удобство использования:
      * Простой и интуитивный интерфейс;
      * Информативные уведомления и подсказки.
   4. Производительность:
      * Приложение должно иметь быстрый доступ к данным;
      * Минимальное время отклика на запросы пользователя.
4. Требования к реализации
   1. Язык программирования: C# и Windows Forms
   2. СУБД: Microsoft SQL Server
5. Требования к документации
   1. Техническое задание на разработку программного модуля.
6. Руководство по стилю
   1. Шрифт: Times New Roman.
   2. Цветовая схема:
      * Основные: White, Black.
      * Второстепенные: DeepPink, DarkRed

Задание 2. Разработка алгоритмов и диаграмм

Словарь данных

Таблица Clients (Клиенты)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Тип данных** | **Notes** | **Примечания** | **Not Null?** |
| PK | ClientID | INT | Синтетический первичный ключ, тип всегда INT (целое) | Автоматически инкрементируемый идентификатор клиента | Y |
|  | FirstName | NVARCHAR(50) | Имя (строка), обязательное поле | Имя клиента | Y |
|  | LastName | NVARCHAR(50) | Фамилия (строка), обязательное поле | Фамилия клиента | Y |
|  | PhoneNumber | NVARCHAR(20) | Телефон (строка), обязательное поле | Номер телефона клиента | Y |

Таблица: Employees (Сотрудники)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Тип данных** | **Примечания** | **Notes** | **Not Null?** |
| PK | EmployeeID | INT | Автоматически инкрементируемый идентификатор сотрудника | Синтетический первичный ключ, тип всегда INT (целое) | **Y** |
|  | FirstName | NVARCHAR(50) | Имя сотрудника | Имя (строка), обязательное поле | **Y** |
|  | LastName | NVARCHAR(50) | Фамилия сотрудника | Фамилия (строка), обязательное поле | **Y** |
|  | Position | NVARCHAR(50) | Должность сотрудника | Должность (строка), обязательное поле | **Y** |

Таблица: Items (Залоговые Предметы)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Типы данных** | **Примечание** | **Notes** | **Not Null?** |
| PK | ItemID | INT | Автоматически инкрементируемый идентификатор предмета | Синтетический первичный ключ, тип всегда INT (целое) | **Y** |
|  | ItemType | NVARCHAR(50) | Тип предмета | Тип предмета (строка), обязательное поле | **Y** |
|  | ItemDescription | NVARCHAR(200) | Описание предмета | Описание предмета (строка), обязательное поле | **Y** |
|  | EstimatedValue | DECIMAL(10, 2) | Оценочная стоимость предмета | Оценочная стоимость (десятичное число), обязательное поле | **Y** |

Таблица: Operations (Залоговые Операции)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Типы данных** | **Примечание** | **Notes** | **Not Null?** |
| PK | OperationID | INT | Автоматически инкрементируемый идентификатор операции | Синтетический первичный ключ, тип всегда INT (целое) | **Y** |
| FK | ClientID | INT | Идентификатор клиента, участвующего в операции | Ссылка на таблицу Clients, обязательное поле | **Y** |
| **FK** | EmployeeID | INT | Идентификатор сотрудника, участвующего в операции | Ссылка на таблицу Employees, обязательное поле | **Y** |
| **FK** | ItemID | INT | Идентификатор предмета, участвующего в операции | Ссылка на таблицу Items, обязательное поле | **Y** |
|  | OperationDate | DATETIME | Дата и время проведения операции | Дата операции (дата и время), обязательное поле по умолчанию GETDATE() | **Y** |
|  | OperationStatus | NVARCHAR(50) | Статус операции (Новая, В процессе, Погашена) | Статус операции (строка), обязательное поле с ограничением CHECK | **Y** |

Таблица: CommentsAndStorage (Комментарии и Хранение)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Тип данных** | **Примечания** | **Notes** | **Not Null?** |
| **PK** | CommentID | INT | Автоматически инкрементируемый идентификатор комментария | Синтетический первичный ключ, тип всегда INT (целое) | **Y** |
| **FK** | OperationID | INT | Идентификатор операции, к которой относится комментарий | Ссылка на таблицу Operations, обязательное поле | **Y** |
|  | CommentText | NVARCHAR(200) | Текст комментария | Текст комментария (строка), обязательное поле | **Y** |
|  | StorageStatus | NVARCHAR(50) | Статус хранения | Статус хранения (строка), обязательное поле | **Y** |

Таблица Users (пользователи)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ключ** | **Название** | **Тип данных** | **Примечание** | **Notice** | **Not Null?** |
| **FK** | UserID | INT | Автоматически инкрементируемый идентификатор пользователя | Синтетический первичный ключ, тип всегда INT (целое) | **Y** |
|  | Username | NVARCHAR(50) | Имя пользователя | Имя пользователя (строка), обязательное поле | **Y** |
|  | Role | NVARCHAR(50) | Роль пользователя (Admin, Employee, Client) | Роль пользователя (строка), обязательное поле с ограничением CHECK | **Y** |
|  | Password | NVARCHAR(255) | Пароль пользователя | Пароль(строка),обязательное поле | **Y** |

UML диаграммы (Рисунок 1-3):



Рисунок 1 - Диаграмма вариантов использования



Рисунок 2 - Диаграммы последовательности;



Рисунок 3 - Диаграмма активности

ER Модель (Рисунок 4):



Рисунок 4 - – ER модель

Создание базы данных:

CREATE TABLE Clients (

ClientID INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

PhoneNumber NVARCHAR(20) NOT NULL

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY(1,1),

FirstName NVARCHAR(50) NOT NULL,

LastName NVARCHAR(50) NOT NULL,

Position NVARCHAR(50) NOT NULL

);

CREATE TABLE Items (

ItemID INT PRIMARY KEY IDENTITY(1,1),

ItemType NVARCHAR(50) NOT NULL,

ItemDescription NVARCHAR(200) NOT NULL,

EstimatedValue DECIMAL(10, 2) NOT NULL

);

CREATE TABLE Operations (

OperationID INT PRIMARY KEY IDENTITY(1,1),

ClientID INT NOT NULL,

EmployeeID INT NOT NULL,

ItemID INT NOT NULL,

OperationDate DATETIME NOT NULL DEFAULT GETDATE(),

OperationStatus NVARCHAR(50) NOT NULL CHECK (OperationStatus IN ('Новая', 'В процессе', 'Погашена')),

CONSTRAINT FK\_Operations\_Clients FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),

CONSTRAINT FK\_Operations\_Employees FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

CONSTRAINT FK\_Operations\_Items FOREIGN KEY (ItemID) REFERENCES Items(ItemID)

);

CREATE TABLE CommentsAndStorage (

CommentID INT PRIMARY KEY IDENTITY(1,1),

OperationID INT NOT NULL,

CommentText NVARCHAR(200) NOT NULL,

StorageStatus NVARCHAR(50) NOT NULL,

CONSTRAINT FK\_CommentsAndStorage\_Operations FOREIGN KEY (OperationID) REFERENCES Operations(OperationID)

);

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY(1,1),

Username NVARCHAR(50) NOT NULL,

Password NVARCHAR(255) NOT NULL,

Role NVARCHAR(50) NOT NULL CHECK (Role IN ('Admin', 'Employee', 'Client')),

);

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY(1,1),

Username NVARCHAR(50) NOT NULL,

Password NVARCHAR(255) NOT NULL,

Role NVARCHAR(50) NOT NULL CHECK (Role IN ('Admin', 'Employee', 'Client')),

);

Заполнение таблиц (Рисунок 5- 9):

INSERT INTO Clients (FirstName, LastName, PhoneNumber)

VALUES

('Иван', 'Иванов', '1234567890'),

('Петр', 'Петров', '9876543210');

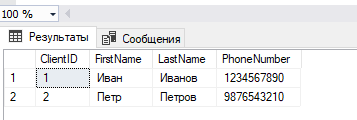


Рисунок 5 - Заполненная таблица Клиенты

INSERT INTO Employees (FirstName, LastName, Position)

VALUES

('Сергей', 'Сергеев', 'Менеджер'),

('Алексей', 'Алексеев', 'Специалист');

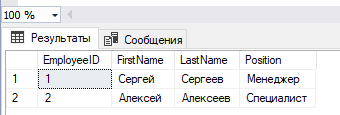


Рисунок 6 - Заполненная таблица Работники

INSERT INTO Items (ItemType, ItemDescription, EstimatedValue)

VALUES

('Ювелирное изделие', 'Золотое кольцо', 10000.00),

('Электроника', 'Смартфон', 50000.00);

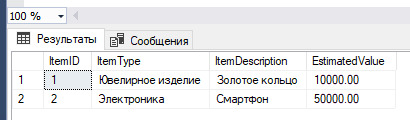


Рисунок 7 - Заполненная таблица Предметы

INSERT INTO Operations (ClientID, EmployeeID, ItemID, OperationStatus)

VALUES

(1, 1, 1, 'Новая'),

(2, 2, 2, 'В процессе');

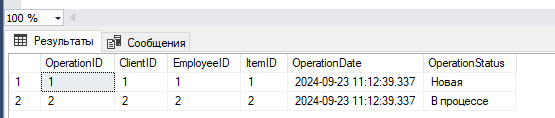


Рисунок 8 - Заполненная таблица Операции

INSERT INTO CommentsAndStorage (OperationID, CommentText, StorageStatus)

VALUES

(1, 'Предмет принят на хранение', 'Хранится'),

(2, 'Предмет находится в процессе оценки', 'На оценке');

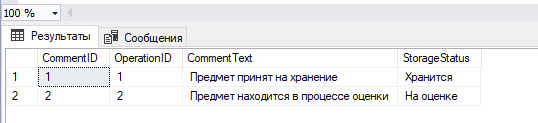


Рисунок 9 - Заполненная таблица Комментарии и Предметы

Задание 3. Макеты

Вайрфрейм макеты (Рисунок 10-12):

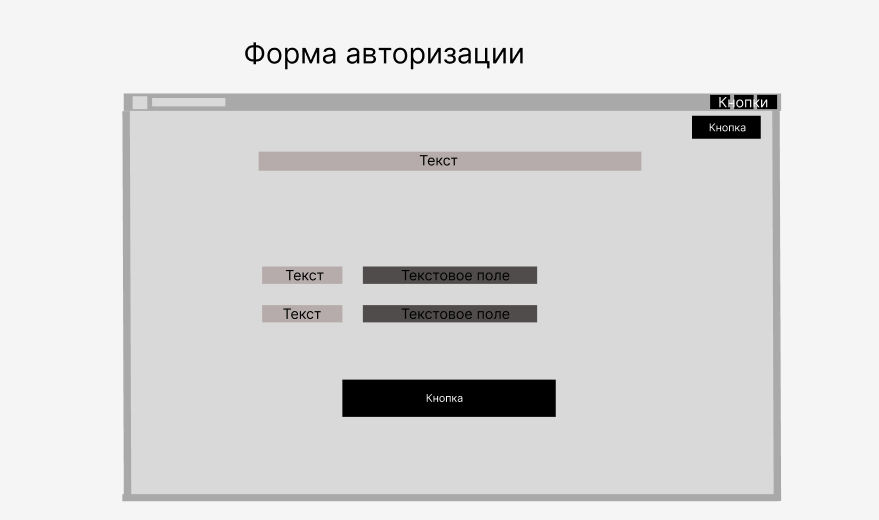


Рисунок 10 - Макет формы авторизации



Рисунок 11 - Макет главной формы

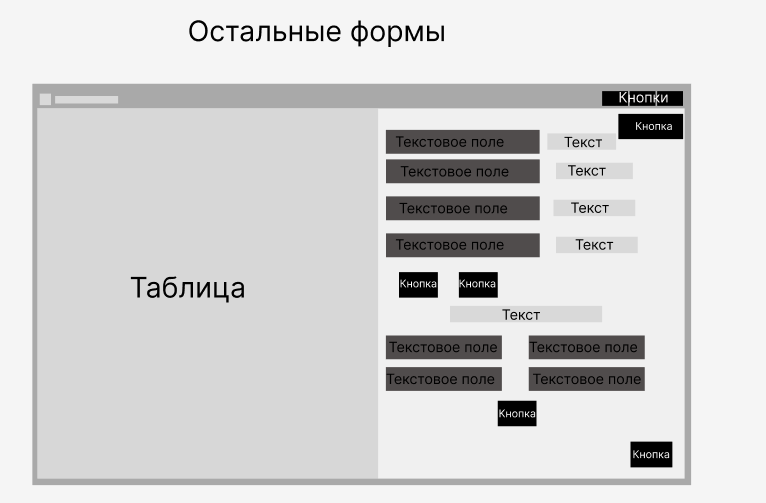


Рисунок 12 - Остальные формы

Мокап макеты (Рисунок 13-15):

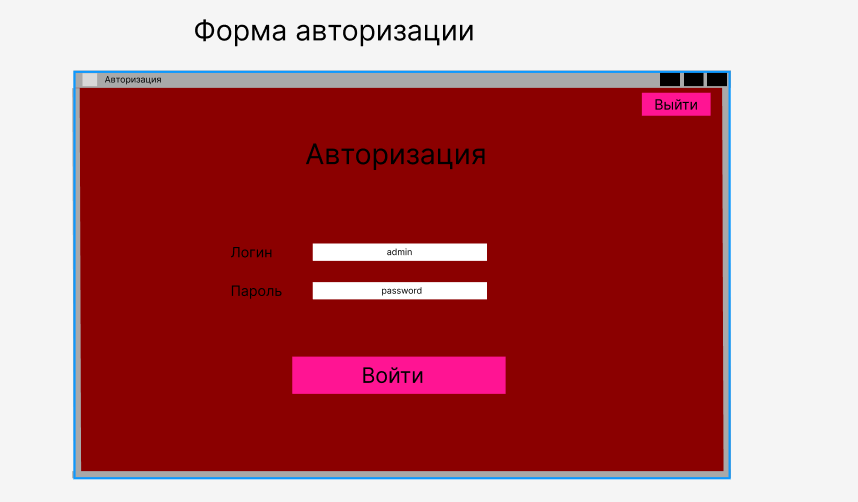


Рисунок 13 - Мокап форма авторизации

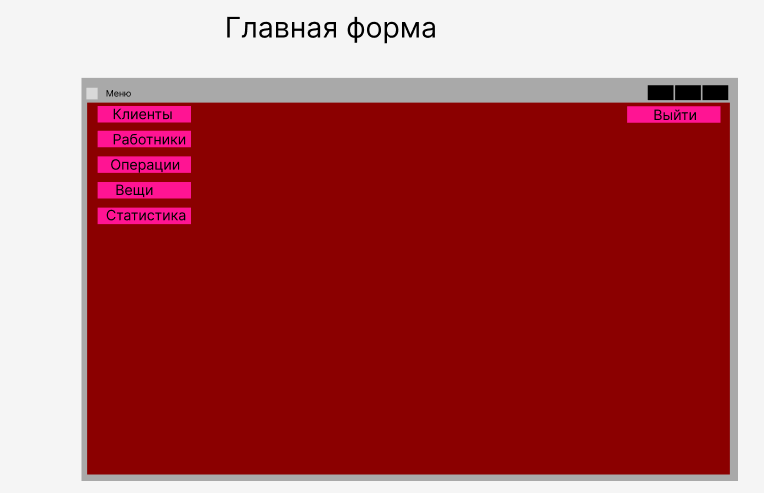


Рисунок 14 - Мокап главная форма

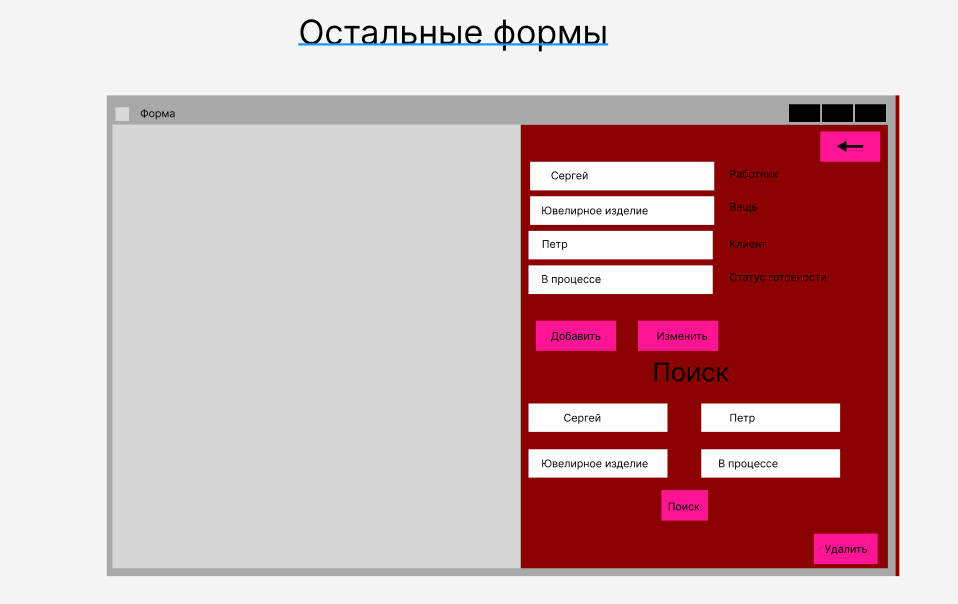


Рисунок 15 - Мокап остальных форм

Карта навигации (Рисунок 16):



Рисунок 16 - Карта навигации

Задание 4. Разработка

Код программы находятся в Приложении А.

При входе в приложение мы видим форму для авторизации, куда вводим Логин “admin” и пароль “123” (Рисунок 17).

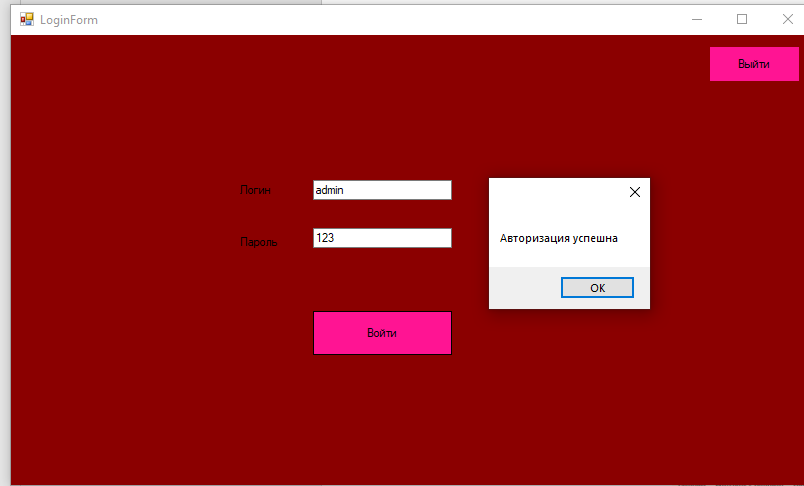


Рисунок 17 - Форма авторизации

После успешной авторизации пользователь попадает в главное меню (Рисунок 18).

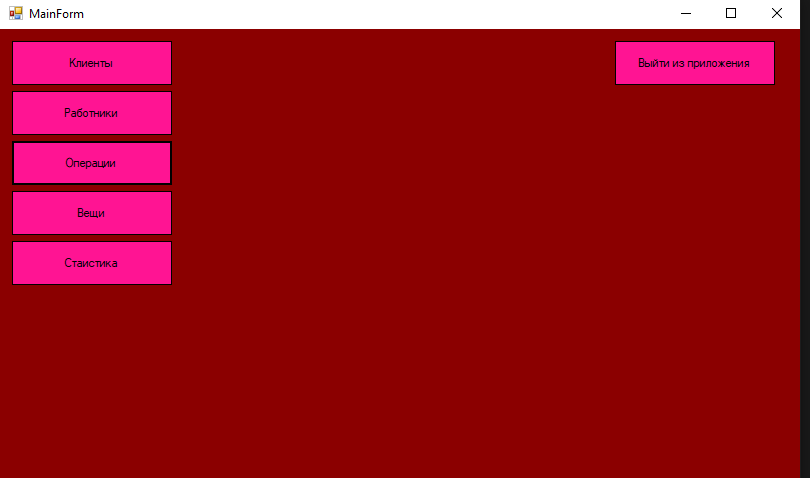


Рисунок 18 - Главная форма

Для вывода определенной таблицы надо нажать на кнопку с соответствующим названием. В нашем случае мы входим в таблицу “Операции”

После нажатия на кнопку “Операции” открывается форма “Операции” (Рисунок 19).

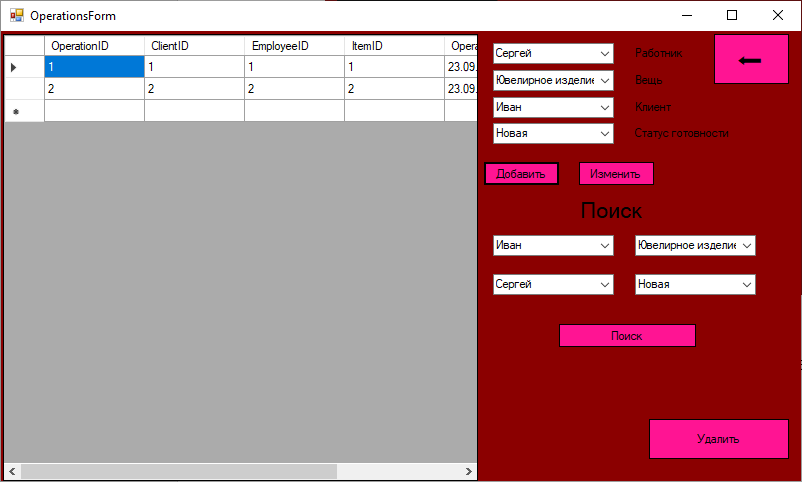


Рисунок 19 - Форма операций

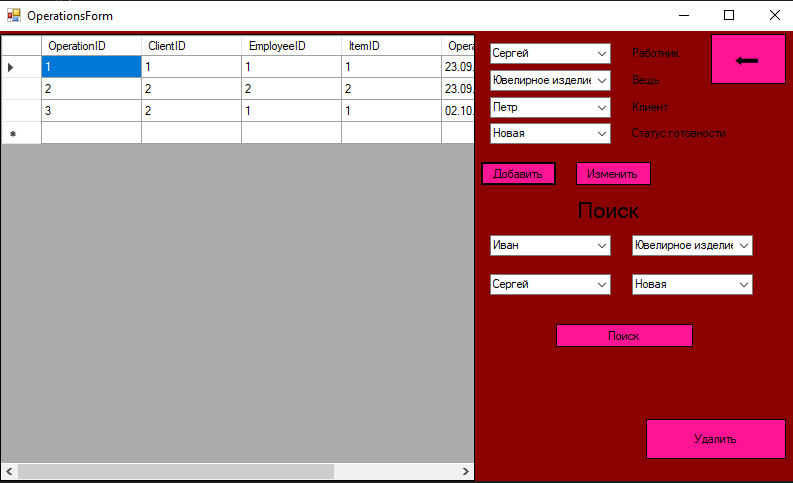
Для добавления данных в таблицу необходимо заполнить все поля, а затем нажать на кнопку “Добавить” (Рисунок 20).  


Рисунок 20 - Добавление новой операции

Для изменения данных в таблицу необходимо заполнить все поля, а затем нажать на кнопку “Изменить” (Рисунок 21).

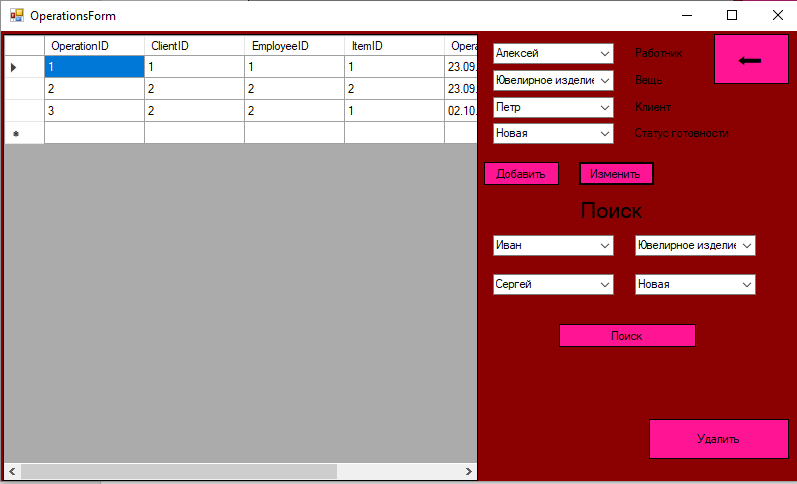


Рисунок 21 - – Изменение операции

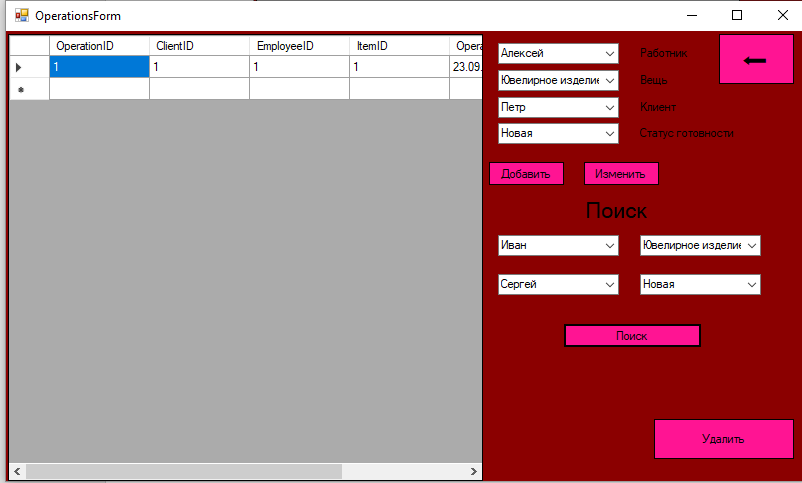
Для поиска записей в таблице необходимо заполнить все поля, а затем нажать на кнопку “Поиск” (Рисунок 22).  


Рисунок 22 - Поиск записей в таблице “Операции”

Для удаления записей в таблице нужно выделить запись и нажать на кнопку “Удалить” (Рисунок 23-24).

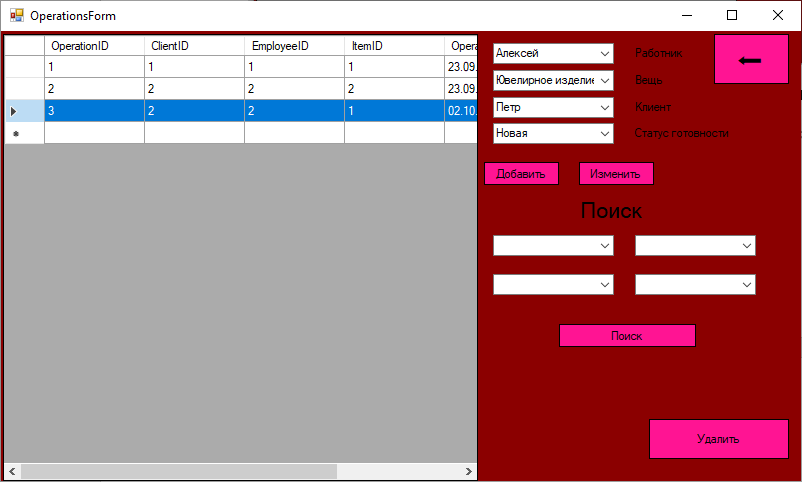


Рисунок 23 - Удаление записи в таблице “Операции”

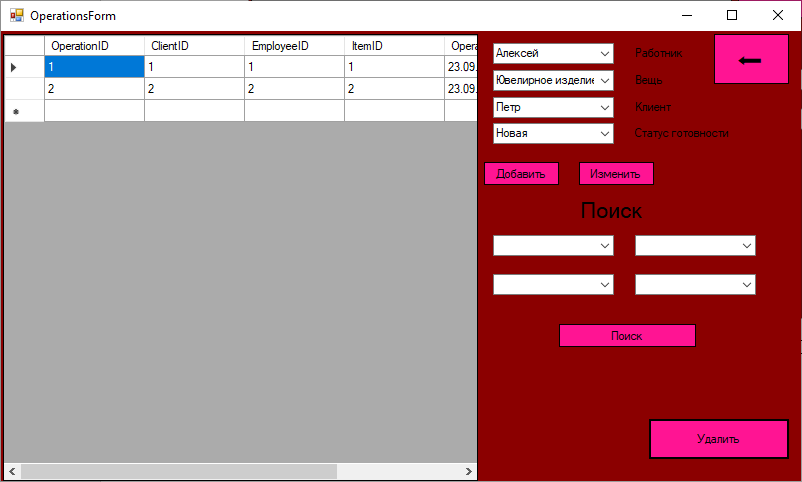


Рисунок 24 - Запись удалена

Далее перейдем в форму “Статистка”. Для выхода в главную форму необходимо нажать на кнопку со значком стрелки (Рисунок 25-26).

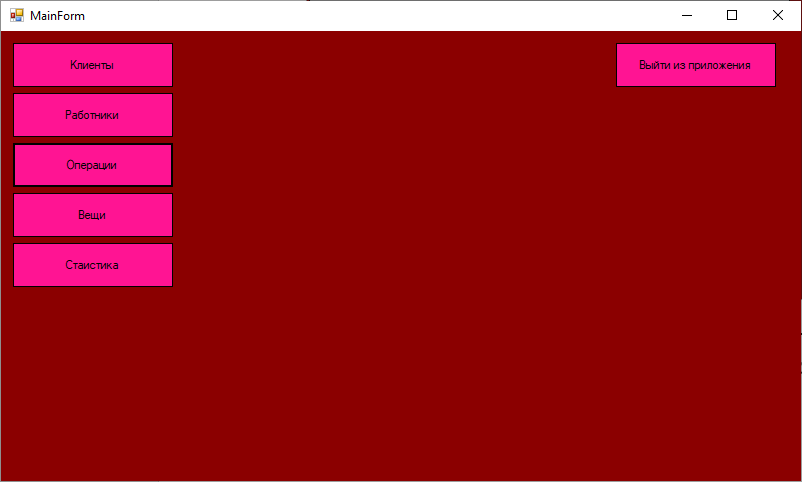


Рисунок 25 - Возвращение в главную форму

Для перехода в форму “Статистка” необходимо нажать на кнопку “Статистика”

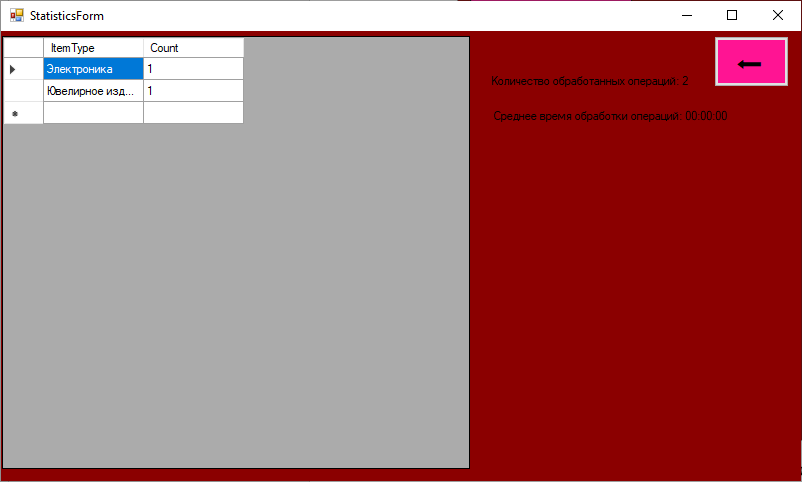


Рисунок 26 - Форма “Статистика”

Задание 5. Тестирование

Тестирование

|  |  |
| --- | --- |
| **Название проекта** | Практика |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Владислав |
| **Дата(ы) теста** | 03.10.2024 |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Пользователь не может авторизироваться с неправильным логином или/и паролем |
| **Краткое изложение теста** | Тест проверяет невозможность авторизации пользователя |
| **Этапы теста** | 1. Написать логин в текстовое поле 2. Написать пароль в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Если записи не существует, выводится ошибка |
| **Тестовые данные** | Логин: «meow», пароль: «2514» |
| **Ожидаемый результат** | Появляется сообщение об ошибке: «Неправильный логин или пароль» |
| **Фактический результат** | Появляется сообщение об ошибке: «Неправильный логин или пароль» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь не авторизован, введён неправильный логин или/и пароль |
|  |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Пользователь авторизовался под администратором |
| **Краткое изложение теста** | Тест проверяет возможность авторизоваться под клиентом |
| **Этапы теста** | 1. Написать логин клиента в текстовое поле 2. Написать пароль клиента в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Успешная авторизация 5. Форма имеет вид, отличающийся от вида формы у администратора |
| **Тестовые данные** | Логин: «admin», пароль: «123» |
| **Ожидаемый результат** | Пользователь попадает на главную форму |
| **Фактический результат** | Пользователь попадает на главную форму |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для клиента |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Пользователь авторизовался под сотрудником |
| **Краткое изложение теста** | Тест проверяет возможность авторизоваться под сотрудником |
| **Этапы теста** | 1. Написать логин сотрудника в текстовое поле 2. Написать пароль сотрудника в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Успешная авторизация 5. Форма имеет вид, отличающийся от вида формы у администратора |
| **Тестовые данные** | Логин: «worker», пароль: «123» |
| **Ожидаемый результат** | Пользователь попадает на главную форму с изменённым видом |
| **Фактический результат** | Пользователь попадает на главную форму с изменённым видом |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для сотрудника |
|  |  |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Пользователь авторизовался под Клиентом |
| **Краткое изложение теста** | Тест проверяет возможность авторизоваться под клиентом |
| **Этапы теста** | 1. Написать логин клиента в текстовое поле 2. Написать пароль клиента в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Успешная авторизация 5. Форма имеет вид, отличающийся от вида формы у администратора |
| **Тестовые данные** | Логин: «client», пароль: «123» |
| **Ожидаемый результат** | Пользователь попадает на главную форму |
| **Фактический результат** | Пользователь попадает на главную форму |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для администратора |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление нового клиента |
| **Краткое изложение теста** | Тест проверяет возможность добавления данных в таблицу Клиенты |
| **Этапы теста** | 1. Авторизоваться под администратором или ролью с соответствующими правами. 2. Перейти на форму клиентов 3. Ввести данные нового клиента: 4. В поле textBoxFirstName ввести имя клиента 5. В поле textBoxLastName ввести фамилию клиента 6. В поле textBoxPhoneNumber ввести номер телефона клиента 7. Нажать кнопку добавления клиента: 8. Нажать кнопку buttonAddOperation для добавления нового клиента в базу данных. 9. Проверить добавление клиента: 10. Проверить, что новый клиент появился в таблице dataGridViewClients. 11. Проверить, что данные нового клиента корректны |
| **Тестовые данные** | Имя: «Иван», фамилия: «Иванов» |
| **Ожидаемый результат** | 1. Новый клиент успешно добавлен в базу данных и отображается в таблице 2. Комбо-боксы для поиска по клиенту обновлены и содержат нового клиента |
| **Фактический результат** | 1. Новый клиент успешно добавлен в базу данных и отображается в таблице 2. Комбо-боксы для поиска по клиенту обновлены и содержат нового клиента |
| **Статус** | Успешно |
| **Предварительное условие** | 1. Пользователь имеет необходимые права доступа для добавления клиентов. 2. Форма клиентов открыта и доступна. |

Юнит-тесты:

Код Юнит-тестов в приложении Б

Результат тестирования (Рисунок 27):

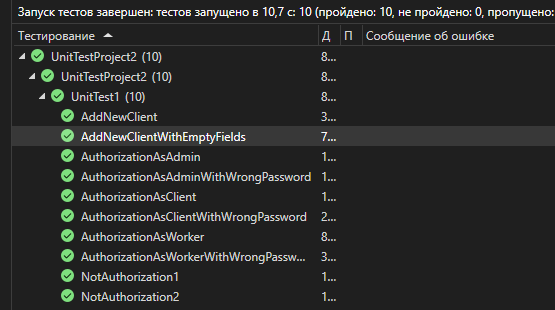


Рисунок 27 - Результат тестирования

Задание 6. GitHub. Оценка проекта

ЗАКЛЮЧЕНИЕ

В данной практике я разработал и протестировал приложение для управления клиентами, которое включает в себя функции добавления, удаления и редактирования информации о клиентах, а также поиск клиентов по имени, фамилии и номеру телефона, и получил опыт в разработке и тестировании приложений, а также улучшил свои навыки программирования и анализа данных, что позволит мне в дальнейшем применить полученные знания и навыки в реальных проектах.

ПРИЛОЖЕНИЕ А

using meow11;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace zalupka3d

{

public partial class LoginForm : Form

{

private SqlConnection connection;

public LoginForm()

{

InitializeComponent();

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

button1.FlatStyle = FlatStyle.Flat;

button1.FlatAppearance.BorderSize = 0;

buttonLogin.FlatStyle = FlatStyle.Flat;

buttonLogin.FlatAppearance.BorderSize = 0;

}

private void LoginForm\_Load(object sender, EventArgs e)

{

}

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

{

// Код для авторизации

string query = "SELECT \* FROM Users WHERE Username = @Username AND Password = @Password ";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Username", textBoxLogin.Text);

command.Parameters.AddWithValue("@Password", textBoxPassword.Text);

SqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

// Авторизация успешна

MessageBox.Show("Авторизация успешна");

string Role = reader["Role"].ToString();

MessageBox.Show("Добро пожаловать, " + Role);

Form1 form1 = new Form1(Role);

form1.Show();

this.Hide();

}

else

{

// Авторизация неуспешна

MessageBox.Show("Неправильный логин или пароль");

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

Код главной формы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using zalupka3d;

namespace meow11

{

public partial class Form1 : Form

{

public string Role;

public Form1(string Role)

{

InitializeComponent();

this.Role = Role;

this.Controls.Add(buttonStatistics);

this.Controls.Add(buttonEmployees);

this.Controls.Add(buttonClients);

this.Controls.Add(buttonOperations);

this.Controls.Add(buttonItems);

buttonStatistics.FlatStyle = FlatStyle.Flat;

buttonEmployees.FlatStyle = FlatStyle.Flat;

buttonClients.FlatStyle = FlatStyle.Flat;

buttonOperations.FlatStyle = FlatStyle.Flat;

buttonItems.FlatStyle = FlatStyle.Flat;

button1.FlatStyle = FlatStyle.Flat;

// Разграничение прав доступа

if (!string.IsNullOrEmpty(Role))

{

switch (Role)

{

case "Admin":

// Администратор

buttonStatistics.Visible = true;

buttonEmployees.Visible = true;

buttonClients.Visible = true;

buttonOperations.Visible = true;

buttonItems.Visible = true;

break;

case "Worker":

// Сотрудник

buttonStatistics.Visible = false;

buttonEmployees.Visible = false;

buttonClients.Visible = false;

buttonOperations.Visible = true;

buttonItems.Visible = true;

break;

case "Client":

// Клиент

buttonStatistics.Visible = false;

buttonEmployees.Visible = false;

buttonClients.Visible = false;

buttonOperations.Visible = true;

buttonItems.Visible = false;

break;

}

}

buttonStatistics.BringToFront();

buttonEmployees.BringToFront();

buttonClients.BringToFront();

buttonOperations.BringToFront();

buttonItems.BringToFront();

this.Refresh();

this.Update();

}

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

{

StatisticsForm statisticsForm = new StatisticsForm();

statisticsForm.Show();

this.Hide();

}

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

{

EmployeesForm employeesForm = new EmployeesForm(Role);

employeesForm.Show();

this.Hide();

}

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

{

ClientsForm clientsForm = new ClientsForm(Role);

clientsForm.Show();

this.Hide();

}

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

{

OperationsForm operationsForm = new OperationsForm(Role);

operationsForm.Show();

this.Hide();

}

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

{

ItemsForm itemsForm = new ItemsForm(Role);

itemsForm.Show();

this.Hide();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

LoginForm loginform = Application.OpenForms["LoginForm"] as LoginForm;

if (loginform != null)

{

loginform.Close();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

Код формы Клиенты

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.VisualStyles;

using System.Net.NetworkInformation;

namespace meow11

{

public partial class ClientsForm : Form

{

private SqlConnection connection;

private SqlDataAdapter adapter;

private DataTable table;

private string Role;

public ClientsForm(string Role)

{

InitializeComponent();

this.Role = Role;

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Clients", connection);

table = new DataTable();

adapter.Fill(table);

dataGridViewClients.DataSource = table;

// Заполнить comboBox для поиска по клиенту

SqlDataAdapter adapterSearchClientsFirstName = new SqlDataAdapter("SELECT ClientID, FirstName FROM Clients", connection);

DataTable tableSearchClientsFirstName = new DataTable();

adapterSearchClientsFirstName.Fill(tableSearchClientsFirstName);

comboBoxSearchFirstName.DataSource = tableSearchClientsFirstName;

comboBoxSearchFirstName.DisplayMember = "FirstName";

comboBoxSearchFirstName.ValueMember = "ClientID";

SqlDataAdapter adapterSearchClientsLastName = new SqlDataAdapter("SELECT ClientID, LastName FROM Clients", connection);

DataTable tableSearchClientsLastName = new DataTable();

adapterSearchClientsLastName.Fill(tableSearchClientsLastName);

comboBoxSearchLastName.DataSource = tableSearchClientsLastName;

comboBoxSearchLastName.DisplayMember = "LastName";

comboBoxSearchLastName.ValueMember = "ClientID";

SqlDataAdapter adapterSearchClientsPhoneNumber = new SqlDataAdapter("SELECT ClientID, PhoneNumber FROM Clients", connection);

DataTable tableSearchClientsPhoneNumber = new DataTable();

adapterSearchClientsPhoneNumber.Fill(tableSearchClientsPhoneNumber);

comboBoxSearchPhoneNumber.DataSource = tableSearchClientsPhoneNumber;

comboBoxSearchPhoneNumber.DisplayMember = "PhoneNumber";

comboBoxSearchPhoneNumber.ValueMember = "ClientID";

buttonDeleteOperation.FlatStyle = FlatStyle.Flat;

buttonAddOperation.FlatStyle = FlatStyle.Flat;

buttonSearch.FlatStyle = FlatStyle.Flat;

button1.FlatStyle = FlatStyle.Flat;

}

private void ClientsForm\_Load(object sender, EventArgs e)

{

}

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

{

}

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

{

// Код для удаления клиента из базы данных

string query = "DELETE FROM Clients WHERE ClientID = @ClientID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ClientID", dataGridViewClients.SelectedRows[0].Cells["ClientID"].Value);

command.ExecuteNonQuery();

// Заполнить comboBox для поиска по клиенту

SqlDataAdapter adapterSearchClients = new SqlDataAdapter("SELECT ClientID, FirstName, LastName, PhoneNumber FROM Clients", connection);

DataTable tableSearchClients = new DataTable();

adapterSearchClients.Fill(tableSearchClients);

comboBoxSearchFirstName.DataSource = tableSearchClients;

comboBoxSearchFirstName.DisplayMember = "FirstName";

comboBoxSearchFirstName.ValueMember = "ClientID";

comboBoxSearchLastName.DataSource = tableSearchClients;

comboBoxSearchLastName.DisplayMember = "LastName";

comboBoxSearchLastName.ValueMember = "ClientID";

comboBoxSearchPhoneNumber.DataSource = tableSearchClients;

comboBoxSearchPhoneNumber.DisplayMember = "PhoneNumber";

comboBoxSearchPhoneNumber.ValueMember = "ClientID";

// Обновить таблицу

table.Clear();

adapter.Fill(table);

dataGridViewClients.DataSource = table;

}

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

{

string query = "INSERT INTO Clients (FirstName, LastName, PhoneNumber) VALUES (@FirstName, @LastName, @PhoneNumber)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@FirstName", textBoxFirstName.Text);

command.Parameters.AddWithValue("@LastName", textBoxLastName.Text);

command.Parameters.AddWithValue("@PhoneNumber", textBoxPhoneNumber.Text);

command.ExecuteNonQuery();

table.Clear();

adapter.Fill(table);

dataGridViewClients.DataSource = table;

}

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

{

string query = "SELECT \* FROM Clients WHERE (FirstName = @FirstName OR @FirstName IS NULL) AND (LastName = @LastName OR @LastName IS NULL) AND (PhoneNumber = @PhoneNumber OR @PhoneNumber IS NULL)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@FirstName", comboBoxSearchFirstName.SelectedItem == null ? (object)DBNull.Value : comboBoxSearchFirstName.SelectedItem.ToString());

command.Parameters.AddWithValue("@LastName", comboBoxSearchLastName.SelectedItem == null ? (object)DBNull.Value : comboBoxSearchLastName.SelectedItem.ToString());

command.Parameters.AddWithValue("@PhoneNumber", comboBoxSearchPhoneNumber.SelectedItem == null ? (object)DBNull.Value : comboBoxSearchPhoneNumber.SelectedItem.ToString());

SqlDataReader reader = command.ExecuteReader();

DataTable tableSearch = new DataTable();

tableSearch.Load(reader);

dataGridViewClients.DataSource = tableSearch;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1(Role);

form1.Show();

this.Close();

}

}

}

Код формы Работники

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace meow11

{

public partial class EmployeesForm : Form

{

private SqlConnection connection;

private SqlDataAdapter adapter;

private DataTable table;

private string Role;

public EmployeesForm(string Role)

{

InitializeComponent();

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Employees", connection);

table = new DataTable();

adapter.Fill(table);

dataGridViewEmployees.DataSource = table;

// Заполнить comboBox для поиска по работнику

SqlDataAdapter adapterSearchEmployeesFirstName = new SqlDataAdapter("SELECT EmployeeID, FirstName FROM Employees", connection);

DataTable tableSearchEmployeesFirstName = new DataTable();

adapterSearchEmployeesFirstName.Fill(tableSearchEmployeesFirstName);

comboBoxSearchFirstName.DataSource = tableSearchEmployeesFirstName;

comboBoxSearchFirstName.DisplayMember = "FirstName";

comboBoxSearchFirstName.ValueMember = "EmployeeID";

SqlDataAdapter adapterSearchEmployeesLastName = new SqlDataAdapter("SELECT EmployeeID, LastName FROM Employees", connection);

DataTable tableSearchEmployeesLastName = new DataTable();

adapterSearchEmployeesLastName.Fill(tableSearchEmployeesLastName);

comboBoxSearchLastName.DataSource = tableSearchEmployeesLastName;

comboBoxSearchLastName.DisplayMember = "LastName";

comboBoxSearchLastName.ValueMember = "EmployeeID";

SqlDataAdapter adapterSearchEmployeesPosition = new SqlDataAdapter("SELECT EmployeeID, Position FROM Employees", connection);

DataTable tableSearchEmployeesPosition = new DataTable();

adapterSearchEmployeesPosition.Fill(tableSearchEmployeesPosition);

comboBoxSearchPosition.DataSource = tableSearchEmployeesPosition;

comboBoxSearchPosition.DisplayMember = "Position";

comboBoxSearchPosition.ValueMember = "EmployeeID";

buttonDeleteOperation.FlatStyle = FlatStyle.Flat;

buttonAddOperation.FlatStyle = FlatStyle.Flat;

buttonSearch.FlatStyle = FlatStyle.Flat;

button1.FlatStyle = FlatStyle.Flat;

}

private void EmployeesForm\_Load(object sender, EventArgs e)

{

}

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

{

// Код для удаления работника из базы данных

string query = "DELETE FROM Employees WHERE EmployeeID = @EmployeeID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@EmployeeID", dataGridViewEmployees.SelectedRows[0].Cells["EmployeeID"].Value);

command.ExecuteNonQuery();

// Обновить таблицу

table.Clear();

adapter.Fill(table);

dataGridViewEmployees.DataSource = table;

}

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

{

string query = "INSERT INTO Employees (FirstName, LastName, Position) VALUES (@FirstName, @LastName, @Position)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@FirstName", textBoxFirstName.Text);

command.Parameters.AddWithValue("@LastName", textBoxLastName.Text);

command.Parameters.AddWithValue("@Position", textBoxPosition.Text);

command.ExecuteNonQuery();

table.Clear();

adapter.Fill(table);

dataGridViewEmployees.DataSource = table;

}

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

{

string query = "SELECT \* FROM Employees WHERE (FirstName = @FirstName OR @FirstName IS NULL) AND (LastName = @LastName OR @LastName IS NULL) AND (Position = @Position OR @Position IS NULL)";

SqlCommand command = new SqlCommand(query, connection);

if (comboBoxSearchFirstName.SelectedIndex != -1)

{

command.Parameters.AddWithValue("@FirstName", comboBoxSearchFirstName.DisplayMember);

}

else

{

command.Parameters.AddWithValue("@FirstName", DBNull.Value);

}

if (comboBoxSearchLastName.SelectedIndex != -1)

{

command.Parameters.AddWithValue("@LastName", comboBoxSearchLastName.DisplayMember);

}

else

{

command.Parameters.AddWithValue("@LastName", DBNull.Value);

}

if (comboBoxSearchPosition.SelectedIndex != -1)

{

command.Parameters.AddWithValue("@Position", comboBoxSearchPosition.DisplayMember);

}

else

{

command.Parameters.AddWithValue("@Position", DBNull.Value);

}

SqlDataReader reader = command.ExecuteReader();

DataTable tableSearch = new DataTable();

tableSearch.Load(reader);

dataGridViewEmployees.DataSource = tableSearch;

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1(Role);

form1.Show();

this.Close();

}

}

}

Код для формы Операции

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.VisualStyles;

using System.Net.NetworkInformation;

namespace meow11

{

public partial class OperationsForm : Form

{

private SqlConnection connection;

private SqlDataAdapter adapter;

private DataTable table;

private string Role;

public OperationsForm(string Role)

{

InitializeComponent();

this.Role = Role;

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Operations", connection);

table = new DataTable();

adapter.Fill(table);

dataGridViewOperations.DataSource = table;

SqlDataAdapter adapterEmployees = new SqlDataAdapter("SELECT EmployeeID, FirstName, LastName FROM Employees", connection);

DataTable tableEmployees = new DataTable();

adapterEmployees.Fill(tableEmployees);

comboBoxEmployee.DataSource = tableEmployees;

comboBoxEmployee.DisplayMember = "FirstName";

comboBoxEmployee.ValueMember = "EmployeeID";

// Заполнить comboBox для выбора предмета

SqlDataAdapter adapterItems = new SqlDataAdapter("SELECT ItemID, ItemType, ItemDescription FROM Items", connection);

DataTable tableItems = new DataTable();

adapterItems.Fill(tableItems);

comboBoxItem.DataSource = tableItems;

comboBoxItem.DisplayMember = "ItemType";

comboBoxItem.ValueMember = "ItemID";

// Заполнить comboBox для выбора клиента

SqlDataAdapter adapterClients = new SqlDataAdapter("SELECT ClientID, FirstName, LastName FROM Clients", connection);

DataTable tableClients = new DataTable();

adapterClients.Fill(tableClients);

comboBoxClient.DataSource = tableClients;

comboBoxClient.DisplayMember = "FirstName";

comboBoxClient.ValueMember = "ClientID";

comboBoxStatus.Items.Add("Новая");

comboBoxStatus.Items.Add("В процессе");

comboBoxStatus.Items.Add("Погашена");

comboBoxStatus.SelectedIndex = 0;

// Заполнить comboBox для поиска по клиенту

SqlDataAdapter adapterSearchClients = new SqlDataAdapter("SELECT ClientID, FirstName, LastName FROM Clients", connection);

DataTable tableSearchClients = new DataTable();

adapterSearchClients.Fill(tableSearchClients);

comboBoxSearchClient.DataSource = tableSearchClients;

comboBoxSearchClient.DisplayMember = "FirstName";

comboBoxSearchClient.ValueMember = "ClientID";

// Заполнить comboBox для поиска по сотруднику

SqlDataAdapter adapterSearchEmployees = new SqlDataAdapter("SELECT EmployeeID, FirstName, LastName FROM Employees", connection);

DataTable tableSearchEmployees = new DataTable();

adapterSearchEmployees.Fill(tableSearchEmployees);

comboBoxSearchEmployee.DataSource = tableSearchEmployees;

comboBoxSearchEmployee.DisplayMember = "FirstName";

comboBoxSearchEmployee.ValueMember = "EmployeeID";

// Заполнить comboBox для поиска по предмету

SqlDataAdapter adapterSearchItems = new SqlDataAdapter("SELECT ItemID, ItemType, ItemDescription FROM Items", connection);

DataTable tableSearchItems = new DataTable();

adapterSearchItems.Fill(tableSearchItems);

comboBoxSearchItem.DataSource = tableSearchItems;

comboBoxSearchItem.DisplayMember = "ItemType";

comboBoxSearchItem.ValueMember = "ItemID";

// Заполнить comboBox для поиска по статусу операции

comboBoxSearchStatus.Items.Add("Новая");

comboBoxSearchStatus.Items.Add("В процессе");

comboBoxSearchStatus.Items.Add("Погашена");

comboBoxSearchStatus.SelectedIndex = 0;

buttonEditOperation.FlatStyle = FlatStyle.Flat;

buttonDeleteOperation.FlatStyle = FlatStyle.Flat;

buttonAddOperation.FlatStyle = FlatStyle.Flat;

buttonSearch.FlatStyle = FlatStyle.Flat;

button1.FlatStyle = FlatStyle.Flat;

}

private void OperationsForm\_Load(object sender, EventArgs e)

{

}

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

{

string query = "UPDATE Operations SET ClientID = @ClientID, EmployeeID = @EmployeeID, ItemID = @ItemID, OperationDate = @OperationDate, OperationStatus = @OperationStatus WHERE OperationID = @OperationID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ClientID", comboBoxClient.SelectedValue);

command.Parameters.AddWithValue("@EmployeeID", comboBoxEmployee.SelectedValue);

command.Parameters.AddWithValue("@ItemID", comboBoxItem.SelectedValue);

command.Parameters.AddWithValue("@OperationDate", DateTime.Now);

command.Parameters.AddWithValue("@OperationStatus", comboBoxStatus.SelectedItem.ToString());

command.Parameters.AddWithValue("@OperationID", dataGridViewOperations.SelectedRows[0].Cells["OperationID"].Value);

command.ExecuteNonQuery();

// Обновить таблицу

table.Clear();

adapter.Fill(table);

dataGridViewOperations.DataSource = table;

}

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

{

// Код для удаления операции из базы данных

string query = "DELETE FROM Operations WHERE OperationID = @OperationID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@OperationID", dataGridViewOperations.SelectedRows[0].Cells["OperationID"].Value);

command.ExecuteNonQuery();

// Обновить таблицу

table.Clear();

adapter.Fill(table);

dataGridViewOperations.DataSource = table;

}

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

{

string query = "INSERT INTO Operations (ClientID, EmployeeID, ItemID, OperationDate, OperationStatus) VALUES (@ClientID, @EmployeeID, @ItemID, @OperationDate, @OperationStatus)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ClientID", comboBoxClient.SelectedValue);

command.Parameters.AddWithValue("@EmployeeID", comboBoxEmployee.SelectedValue);

command.Parameters.AddWithValue("@ItemID", comboBoxItem.SelectedValue);

command.Parameters.AddWithValue("@OperationDate", DateTime.Now);

command.Parameters.AddWithValue("@OperationStatus", "Новая");

command.ExecuteNonQuery();

table.Clear();

adapter.Fill(table);

dataGridViewOperations.DataSource = table;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1(Role);

form1.Show();

this.Close();

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

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

{

string query = "SELECT \* FROM Operations WHERE (ClientID = @ClientID OR @ClientID IS NULL) AND (EmployeeID = @EmployeeID OR @EmployeeID IS NULL) AND (ItemID = @ItemID OR @ItemID IS NULL) AND (OperationStatus = @OperationStatus OR @OperationStatus IS NULL)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ClientID", comboBoxSearchClient.SelectedValue == null ? (object)DBNull.Value : comboBoxSearchClient.SelectedValue);

command.Parameters.AddWithValue("@EmployeeID", comboBoxSearchEmployee.SelectedValue == null ? (object)DBNull.Value : comboBoxSearchEmployee.SelectedValue);

command.Parameters.AddWithValue("@ItemID", comboBoxSearchItem.SelectedValue == null ? (object)DBNull.Value : comboBoxSearchItem.SelectedValue);

command.Parameters.AddWithValue("@OperationStatus", comboBoxSearchStatus.SelectedItem == null ? (object)DBNull.Value : comboBoxSearchStatus.SelectedItem.ToString());

SqlDataReader reader = command.ExecuteReader();

DataTable tableSearch = new DataTable();

tableSearch.Load(reader);

dataGridViewOperations.DataSource = tableSearch;

}

}

}

Код для формы Вещи

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace meow11

{

public partial class ItemsForm : Form

{

private SqlConnection connection;

private SqlDataAdapter adapter;

private DataTable table;

private string Role;

public ItemsForm(string Role)

{

InitializeComponent();

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Items", connection);

table = new DataTable();

adapter.Fill(table);

dataGridViewItems.DataSource = table;

SqlDataAdapter adapterSearchItemType = new SqlDataAdapter("SELECT DISTINCT ItemType FROM Items", connection);

DataTable tableSearchItemType = new DataTable();

adapterSearchItemType.Fill(tableSearchItemType);

comboBoxSearchItemType.DataSource = tableSearchItemType;

comboBoxSearchItemType.DisplayMember = "ItemType";

comboBoxSearchItemType.ValueMember = "ItemType";

SqlDataAdapter adapterItems = new SqlDataAdapter("SELECT ItemID, ItemType, ItemDescription, EstimatedValue FROM Items", connection);

DataTable tableItems = new DataTable();

adapterItems.Fill(tableItems);

textBoxItemType.DataBindings.Add("Text", tableItems, "ItemType");

textBoxItemDescription.DataBindings.Add("Text", tableItems, "ItemDescription");

textBoxEstimatedValue.DataBindings.Add("Text", tableItems, "EstimatedValue");

SqlDataAdapter adapterSearchItemDescription = new SqlDataAdapter("SELECT DISTINCT ItemDescription FROM Items", connection);

DataTable tableSearchItemDescription = new DataTable();

adapterSearchItemDescription.Fill(tableSearchItemDescription);

comboBoxSearchItemDescription.DataSource = tableSearchItemDescription;

comboBoxSearchItemDescription.DisplayMember = "ItemDescription";

comboBoxSearchItemDescription.ValueMember = "ItemDescription";

buttonAddOperation.FlatStyle = FlatStyle.Flat;

buttonSearch.FlatStyle = FlatStyle.Flat;

buttonDeleteOperation.FlatStyle = FlatStyle.Flat;

button1.FlatStyle = FlatStyle.Flat;

}

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

{

string query = "INSERT INTO Items (ItemType, ItemDescription, EstimatedValue) VALUES (@ItemType, @ItemDescription, @EstimatedValue)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ItemType", textBoxItemType.Text);

command.Parameters.AddWithValue("@ItemDescription", textBoxItemDescription.Text);

command.Parameters.AddWithValue("@EstimatedValue", decimal.Parse(textBoxEstimatedValue.Text));

command.ExecuteNonQuery();

// Update the table

table.Clear();

adapter.Fill(table);

dataGridViewItems.DataSource = table;

}

private void ItemsForm\_Load(object sender, EventArgs e)

{

}

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

{

string query = "SELECT \* FROM Items WHERE (ItemType = @ItemType OR @ItemType IS NULL) AND (ItemDescription = @ItemDescription OR @ItemDescription IS NULL)";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ItemType", comboBoxSearchItemType.SelectedValue == null ? (object)DBNull.Value : comboBoxSearchItemType.SelectedValue);

command.Parameters.AddWithValue("@ItemDescription", comboBoxSearchItemDescription.SelectedValue == null ? (object)DBNull.Value : comboBoxSearchItemDescription.SelectedValue);

SqlDataReader reader = command.ExecuteReader();

DataTable tableSearch = new DataTable();

tableSearch.Load(reader);

reader.Close();

dataGridViewItems.DataSource = tableSearch;

}

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

{

string query = "DELETE FROM Items WHERE ItemID = @ItemID";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@ItemID", dataGridViewItems.SelectedRows[0].Cells["ItemID"].Value);

command.ExecuteNonQuery();

// Update the table

table.Clear();

adapter.Fill(table);

dataGridViewItems.DataSource = table;

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1(Role);

form1.Show();

this.Close();

}

}

}

Код для формы Статистика

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace meow11

{

public partial class StatisticsForm : Form

{

private SqlConnection connection;

private SqlDataAdapter adapter;

private DataTable table;

private string Role;

public StatisticsForm()

{

InitializeComponent();

this.Load += new EventHandler(StatisticsForm\_Load);

connection = new SqlConnection("Data Source=adclg1;Initial Catalog=Praktika\_filimonov;Integrated Security=True;Encrypt=False");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Operations", connection);

table = new DataTable();

adapter.Fill(table);

// Количество обработанных операций

int countOperations = table.Rows.Count;

labelCountOperations.Text = "Количество обработанных операций: " + countOperations.ToString();

// Среднее время обработки операций

DateTime startTime = DateTime.Now;

DateTime endTime = DateTime.Now;

long totalTicks = 0;

foreach (DataRow row in table.Rows)

{

startTime = Convert.ToDateTime(row["OperationDate"]);

endTime = Convert.ToDateTime(row["OperationDate"]);

totalTicks += (endTime - startTime).Ticks;

}

TimeSpan averageTime = TimeSpan.FromTicks(totalTicks / table.Rows.Count);

labelAverageTime.Text = "Среднее время обработки операций: " + averageTime.ToString();

// Статистика по типам залоговых предметов

SqlDataAdapter adapterItems = new SqlDataAdapter("SELECT ItemType, COUNT(\*) AS Count FROM Items GROUP BY ItemType", connection);

DataTable tableItems = new DataTable();

adapterItems.Fill(tableItems);

dataGridViewStatistics.DataSource = tableItems;

}

private void StatisticsForm\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

Form1 form1 = new Form1(Role);

form1.Show();

this.Close();

}

}

}

ПРИЛОЖЕНИЕ Б

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using Moq;

using zalupka3d;

using meow11;

namespace UnitTestProject2

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void NotAuthorization1()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "meow";

loginForm.PasswordText = "2514";

var exception = Assert.ThrowsException<Exception>(() => loginForm.buttonLogin\_Click(new object(), new EventArgs()));

Assert.AreEqual("Неправильный логин или пароль", exception.Message);

}

[TestMethod]

public void NotAuthorization2()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "123";

loginForm.PasswordText = "123";

var exception = Assert.ThrowsException<Exception>(() => loginForm.buttonLogin\_Click(new object(), new EventArgs()));

Assert.AreEqual("Неправильный логин или пароль", exception.Message);

}

[TestMethod]

public void AuthorizationAsAdmin()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "admin";

loginForm.PasswordText = "123";

loginForm.buttonLogin\_Click(new object(), new EventArgs());

Form1 form1 = new Form1("admin"); // создаем экземпляр формы Form1

Assert.IsTrue(loginForm.Visible == false); // Форма должна быть скрыта после авторизации

}

[TestMethod]

public void AuthorizationAsAdminWithWrongPassword()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "admin";

loginForm.PasswordText = "wrongpassword";

var exception = Assert.ThrowsException<Exception>(() => loginForm.buttonLogin\_Click(new object(), new EventArgs()));

Assert.AreEqual("Неправильный логин или пароль", exception.Message);

}

[TestMethod]

public void AuthorizationAsWorker()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "worker";

loginForm.PasswordText = "123";

loginForm.buttonLogin\_Click(new object(), new EventArgs());

Form1 form1 = new Form1("worker");

Assert.IsTrue(loginForm.Visible == false); // Форма должна быть скрыта после авторизации

}

[TestMethod]

public void AuthorizationAsWorkerWithWrongPassword()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "worker";

loginForm.PasswordText = "wrongpassword";

var exception = Assert.ThrowsException<Exception>(() => loginForm.buttonLogin\_Click(new object(), new EventArgs()));

Assert.AreEqual("Неправильный логин или пароль", exception.Message);

}

[TestMethod]

public void AuthorizationAsClient()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "client";

loginForm.PasswordText = "123";

loginForm.buttonLogin\_Click(new object(), new EventArgs());

Form1 form1 = new Form1("client");

Assert.IsTrue(loginForm.Visible == false); // Форма должна быть скрыта после авторизации

}

[TestMethod]

public void AuthorizationAsClientWithWrongPassword()

{

LoginForm loginForm = new LoginForm();

loginForm.LoginText = "client";

loginForm.PasswordText = "wrongpassword";

var exception = Assert.ThrowsException<Exception>(() => loginForm.buttonLogin\_Click(new object(), new EventArgs()));

Assert.AreEqual("Неправильный логин или пароль", exception.Message);

}

[TestMethod]

public void AddNewClient()

{

ClientsForm clientsForm = new ClientsForm("admin");

clientsForm.TextBoxFirstNameText = "Иван";

clientsForm.TextBoxLastNameText = "Иванов";

clientsForm.TextBoxPhoneNumberText = "1234567890";

clientsForm.buttonAddOperation\_Click(new object(), new EventArgs());

Assert.IsTrue(clientsForm.DataGridViewClients.Rows.Count > 0);

Assert.AreEqual("Иван", clientsForm.DataGridViewClients.Rows[0].Cells["FirstName"].Value);

Assert.AreEqual("Иванов", clientsForm.DataGridViewClients.Rows[0].Cells["LastName"].Value);

}

[TestMethod]

public void AddNewClientWithEmptyFields()

{

ClientsForm clientsForm = new ClientsForm("admin");

int initialRowsCount = clientsForm.DataGridViewClients.Rows.Count;

clientsForm.TextBoxFirstNameText = "";

clientsForm.TextBoxLastNameText = "";

clientsForm.TextBoxPhoneNumberText = "";

clientsForm.buttonAddOperation\_Click(new object(), new EventArgs());

Assert.IsTrue(clientsForm.DataGridViewClients.Rows.Count == initialRowsCount);

}

}

**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 42919/2 группы

Филимонов Владислав Анатольевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**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 Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «05» октября 2024 г.