[Работа 1 2](#_Toc178765724)

[Работа 2 6](#_Toc178765725)

[Работа 3 13](#_Toc178765726)

[Работа 4 19](#_Toc178765727)

[Работа 5 24](#_Toc178765728)

Задание 1. Техническое задание

Описание предметной области

Основной целью данного модуля является создание регистрации пользователей и отслеживание книг в библиотеке. Модуль позволяет сотрудникам библиотеки прозрачно и эффективно отслеживать статус книг.

Разработка программного модуля для регистрации пользователей и отслеживания книг - это процесс создания программного продукта, который позволяет сотрудникам библиотеки статус книг.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для отслеживания и редактирования информации о книгах, что помогает вести учет и обеспечивает бесперебойную работу в библиотеке.

Программный модуль позволяет библиотеке повысить эффективность и точность статуса, сократить время поиска книг и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1. Регистрация пользователя: это информация, предоставленная клиентом. Пользователь создает аккаунт. Аккаунт содержит id клиента, его ФИО. Клиент может отредактировать свой личный кабинет.

2. Учет статуса книги: этот процесс включает статус книги в системе учёта. Важными аспектами учета являются присвоение статуса книги, ее данные (id книги, название книги, количество страниц, издатель), сохранение информации о статусе.

3. Обработка статуса книги: процесс, включающий изменение статуса книги. В процессе обработки будут установлены сроки бронирования книги.

4. Бронирование книги: фактическое выполнение бронирования книги. На этом этапе библиотекарь изменяет статус книги, а также добавляет книгу в карту читателя. При необходимости срок держания книги можно продлить.

5. Отчётность и информирование: важной составляющей учёта является изменение статуса книги. После возвращения книги ее статус изменяется.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса учёта статуса книг. Важно отслеживать и анализировать сроки возвращения книг которые могут помочь в оптимизации и улучшении процесса.

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для отслеживания книг в библиотеке.

1.2. Заказчик: Библиотека “BookDaBook”

1.3. Исполнитель: Компания “Grzzz”

2. Функциональные требования

2.1. Возможность добавления пользователей в базу данных с указанием следующих параметров:

- id пользователя;

- ФИО пользователя;

- Номер телефона пользователя;

- Название книги;

- Дата взятия книги.

2.2. Возможность редактирования пользователя:

- Изменение ФИО пользователя;

- Изменение номера телефона пользователя;

- Изменение даты взятия книги (продление);

- Изменение взятой книги.

2.3. Возможность отслеживания статуса книг:

- Отображение списка забронированных книг и книг на реставрации;

- Получение уведомлений об окончании срока;

- Поиск забронированной книги по названию или по параметрам.

2.4. Возможность назначения ответственных за выполнение работ:

- Отслеживание сроков взятия книги и получение уведомлений о их завершении.

2.5. Расчет статистики работы отдела обслуживания:

- Статистика забронированных книг;

- Статистика должников книг.

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на бронирование книги.

4. Требования к реализации

4.1. Язык программирования: C#

4.2. СУБД: mySQL

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Times New Roman

6.2. Цветовая схема: Монохромная палитра

Основные цвета: #166DC6, #EDEDF6

Дополнительные: #271663

Задание 2. Диаграммы. БД



Рисунок 1 – Диаграмма вариантов использования



Рисунок 2 – Диаграмма последовательности



Рисунок 3 – Диаграмма активности



Рисунок 4 – ER-диаграмма

Таблица 1 – Таблица всех людей в бд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Person | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | id | INTEGER | Y | Auto Increment |
|  | name | NVARCHAR (30) | Y |  |
|  | phone\_number | NVARCHAR (16) | Y |  |
|  | password | NVARCHAR(30) | Y |  |
| FK | role\_id | INTEGER | Y |  |

Таблица 2 – Таблица доступности книги

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Availability | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | id | INTEGER | Y | Auto Increment |
|  | status | BIT | Y |  |

Таблица 3 – Таблица доступности книги

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Book | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | id | INTEGER | Y | Auto Increment |
|  | name | NVARCHAR(30) | Y |  |
|  | year | INTEGER | Y |  |
|  | publisher | NVARCHAR(30) | Y |  |
| FK | status\_id | INTEGER | Y |  |

Таблица 4 – Таблица ролей людей в бд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Worker | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| PK | id | INTEGER | Y |  |
|  | role\_name | NVARCHAR (30) | Y |  |

Таблица 5 – Таблица карточек читателя

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cards | | | | |
| KEY | **FIELD NAME** | **DATA TYPE / FIELD SIZE** | **REQUIRED?** | **NOTES** |
| FK | book\_id | INTEGER | Y |  |
| FK | reader\_id | INTEGER | Y |  |
|  | get\_date | DATE | Y |  |
|  | return\_date | DATE | Y |  |
| FK | worder\_id | INTEGER | Y |  |

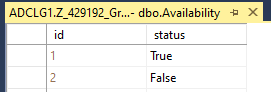


Рисунок 5 – Таблица доступности книги

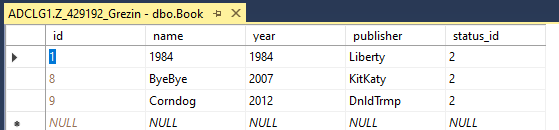


Рисунок 6 – Таблица книг

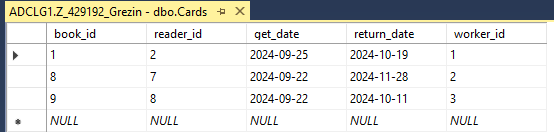


Рисунок 7- Таблица карточки читателя

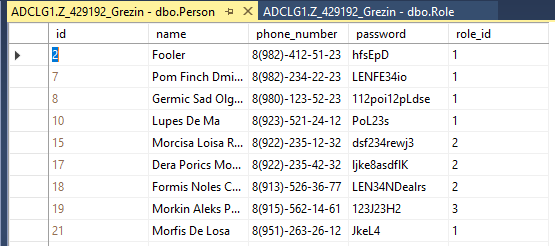


Рисунок 8 – Таблица людей

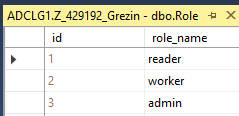


Рисунок 9 – Таблица ролей

Задание 3. Макеты

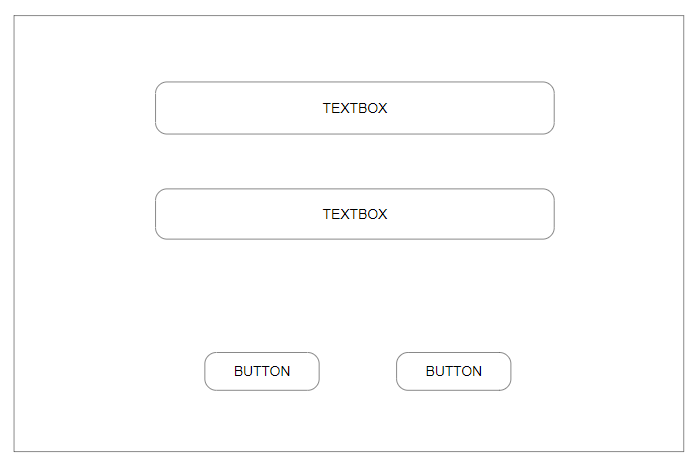


Рисунок 10 - Wireframe макет авторизации/входа



Рисунок 11 – Wireframe макет списка книг

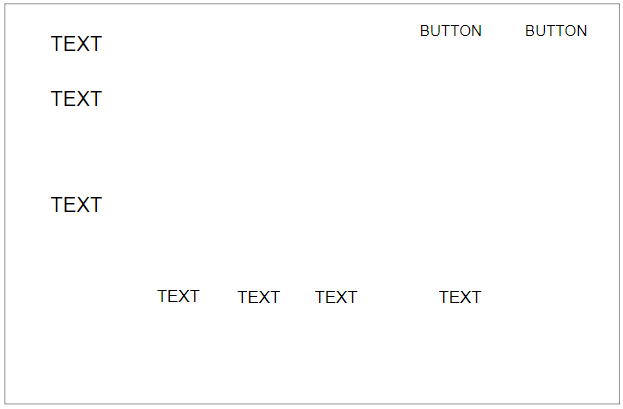


Рисунок 12 – Wireframe макет личного кабинета

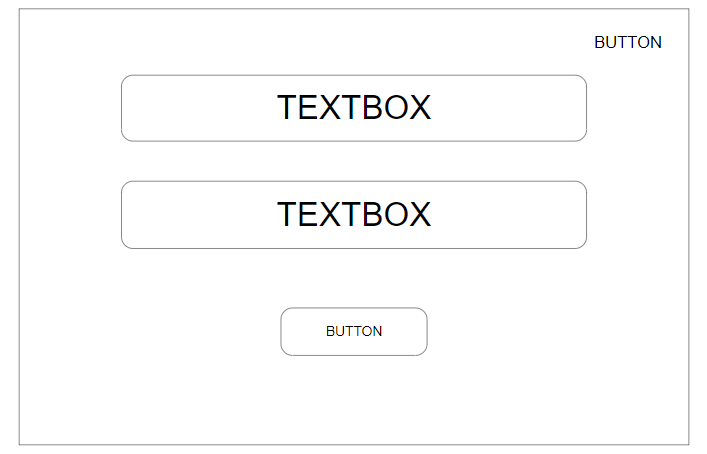


Рисунок 13 – Wireframe макет изменения имени пользователя

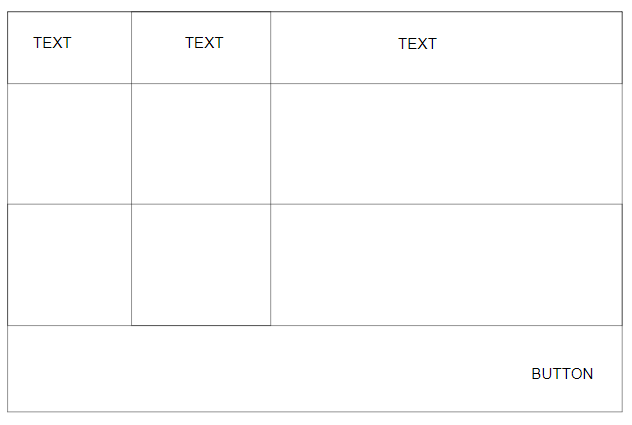


Рисунок 14 – Wireframe макет изменения данных всех людей в бд

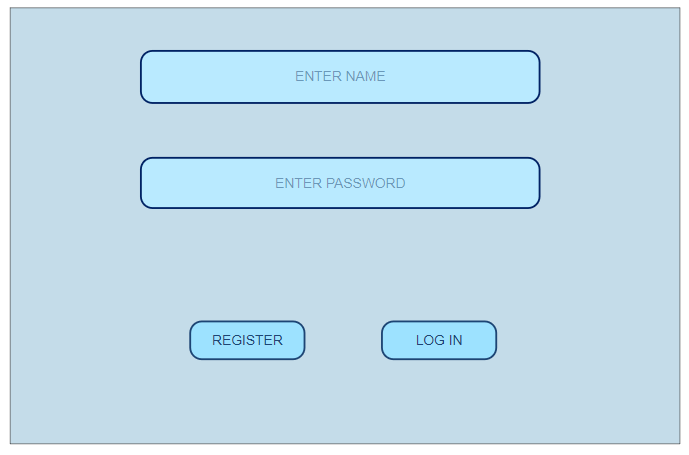


Рисунок 15 – Mockup макет авторизации/входа

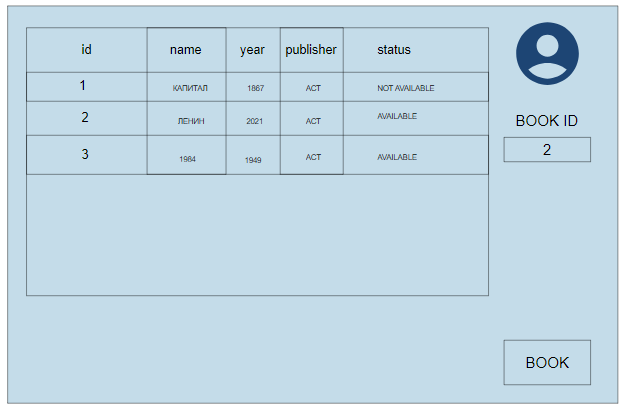


Рисунок 16 – Mockup макет списка книг

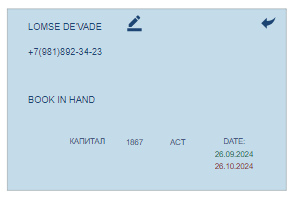


Рисунок 17 – Mockup макет личного кабинета пользователя

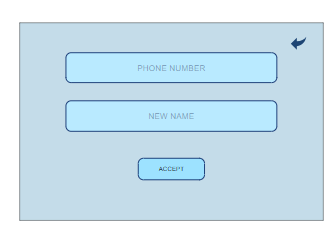


Рисунок 18 – Mockup макет изменения имени пользователя

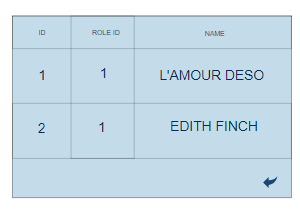


Рисунок 19 – Mockup макет изменения данных всех людей в БД

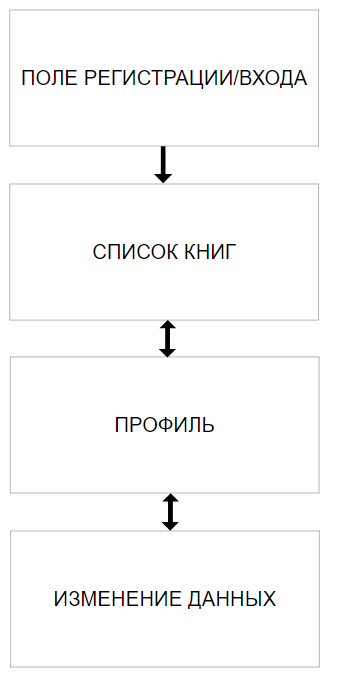


Рисунок 20 – Карта навигации

<https://app.moqups.com/9hRkSRVHyZvoatVTImTONu1k64mz5Z3g/edit/page/ad64222d5>

Задание 4. Разработка

Пользователь:

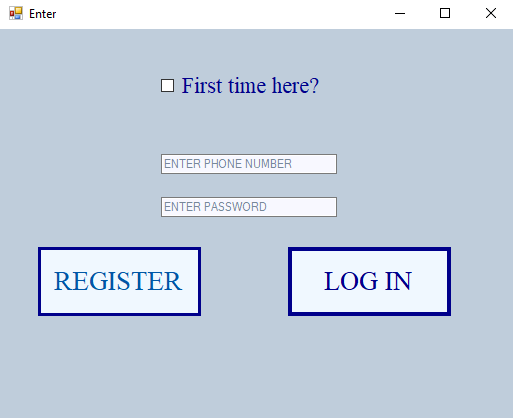
****

Рисунок 21 - Форма входа

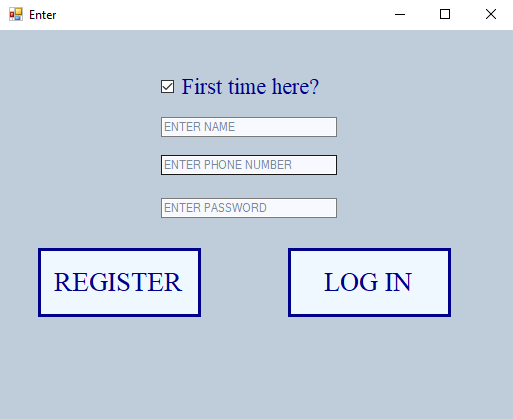


Рисунок 22 - Форма регистрации

При нажатии на checkbox становится доступна кнопка регистрации. При вводе корректных данных и нажатии на кнопку “REGISTER” пользователь добавляется в БД. При вводе существующих в базе данных пользователя и нажатии на кнопку “LOG IN” пользователь впускается в приложение.

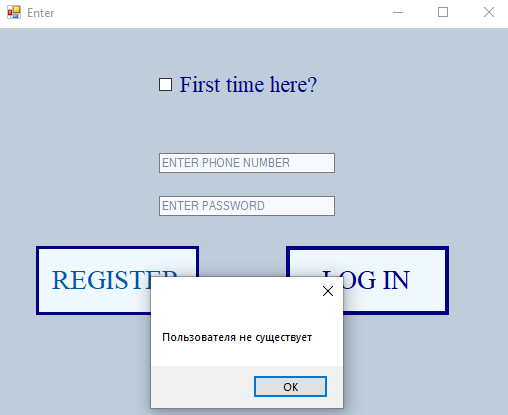


Рисунок 23 - Ввод неправильных данных при входе

При вводе некорректных данных пользователю высветится ошибка (Рисунок 23).

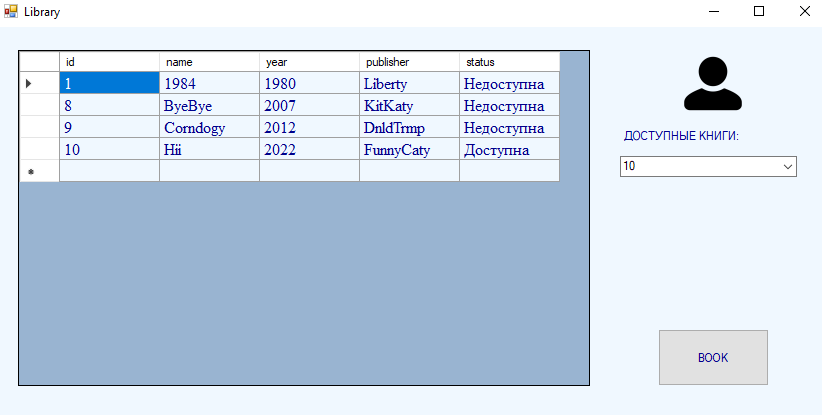


Рисунок 24 - Список книг

При прохождении формы входа (Рисунок 21) выводится список книг и выпадающий список с книгами, которые можно забронировать (status = Доступна), выбрав id книги из выпадающего списка и нажав на кнопку “BOOK”. Справа сверху находится кнопка, при нажатии на которую можно перейти в личный кабинет (Рисунок 25).



Рисунок 25 - Личный кабинет

При открытии личного кабинета (Рисунок 25) высветятся личные данные пользователя и информация о книге у него на руках. При нажатии на кнопку, находящуюся справа от имени можно изменить данные пользователя. При нажатии на кнопку слева сверху можно вернуться на предыдущую форму (Рисунок 24).

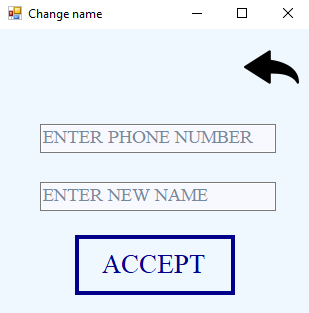


Рисунок 26 - Форма изменения данных пользователя(имени)

В форме изменения данных (Рисунок 26) можно изменить имя пользователя, введя корректный номер телефона.

Работник:

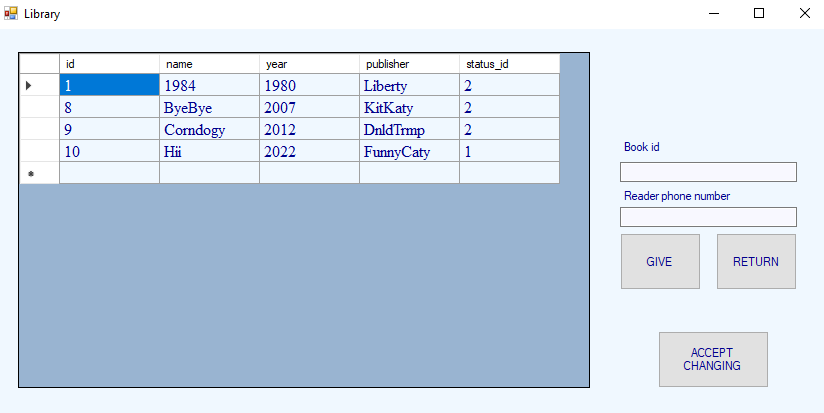


Рисунок 27 - Форма со списком книг

При вводе корректных данных открывается форма со списком книг (Рисунок 27), который можно изменить (изменить в таблице(удалить, добавить, изменить доступные ячейки) и нажать на кнопку “ACCEPT CHANGING”). Также можно присвоить книгу пользователю и оформить возвращение книги, заполнив соответствующие поля и нажав на соответствующую кнопку.

Администратор

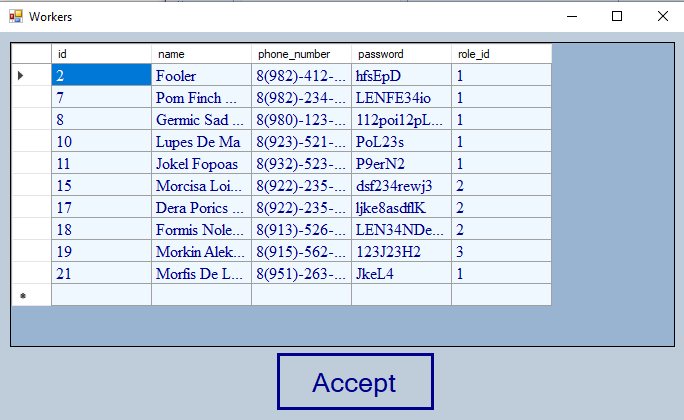


Рисунок 28 - Форма с данными людей

Форма с изменением, добавлением и удалением всех людей в базе (Рисунок 28). Изменив данные в таблице(добавив, удалив, изменив доступные ячейки) и нажав на кнопку “ACCEPT”, они изменятся в БД.

Задание 5. Тестирование

Таблица 6 – Аннотация к тестам

|  |  |
| --- | --- |
| **Название проекта** | BookDaBook |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Грезин Данил Максимович |
| **Дата(ы) теста** | **02.10.2024** |

Таблица 7 – Тестовый пример 1

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Подтвердите страницу авторизации с действительным номером телефона и паролем. |
| **Краткое изложение теста** | При правильном заполнении полей осуществляется вход в систему. |
| **Этапы теста** | 1. Заполнение соответствующих текстовых полей номером телефона и паролем. 2. Нажатие на кнопку “LOG IN”. |
| **Тестовые данные** | Номер телефона: 8(999)-999-99-99  Имя: Angelo Demuas |
| **Ожидаемый результат** | При вводе корректных данных открывается форма, соответствующая роли пользователя.  При вводе некорректных данных выводится ошибка о неправильно введенных данных. |
| **Фактический результат** | Открытие формы, в зависимости от роли человека, вводившего данные |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен быть зарегистрирован в приложении. |
| **Постусловие** | - |
| **Примечания/комментарии** | Пользователь авторизуется и в зависимости от его данных выводится форма, соответствующая его роли (читатель, работник, администратор). |

Таблица 8 – Тестовый пример 2

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Блокировка кнопки регистрации. |
| **Краткое изложение теста** | При незаполненных полях регистрации, кнопка регистрации блокируется. |
| **Этапы теста** | Не заполнять поля регистрации. |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Кнопка недоступна. |
| **Фактический результат** | Кнопка недоступна |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | При заполнении полей, кнопка регистрации становится доступна. |

Таблица 9 – Тестовый пример 3

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Бронирование книги. |
| **Краткое изложение теста** | Пользователь бронирует доступную книгу. |
| **Этапы теста** | 1. Пользователь выбирает id доступной книги 2. Пользователь нажимает “BOOK” |
| **Тестовые данные** | id книги: 10 |
| **Ожидаемый результат** | Книга станет недоступной. |
| **Фактический результат** | Книга станет недоступной. |
| **Статус** | Зачет |
| **Предварительное условие** | Книга является доступной. У пользователя нет книг на руках. |
| **Постусловие** | Книга является недоступной. |
| **Примечания/комментарии** | Пользователь бронирует доступную книгу, из-за чего она становится недоступной. |

Таблица 10 – Тестовый пример 4

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Возвращение книги в библиотеку. |
| **Краткое изложение теста** | При вводе id книги и номера телефона пользователя и нажатии кнопки “RETURN”, книга удаляется из личного кабинета пользователя и становится доступной. |
| **Этапы теста** | 1. Ввод id книги и номера телефона пользователя, который возвращает книгу. 2. Нажатие на кнопку “RETURN”. |
| **Тестовые данные** | id книги: 10  номер телефона пользователя: 8(999)-999-99-99 |
| **Ожидаемый результат** | Книга удаляется из карточки читателя и становится доступной. |
| **Фактический результат** | Книга удаляется из карточки читателя и становится доступной. |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь существует в базе данных. |
| **Постусловие** | Книга удаляется из карты пользователя и становится доступной. |
| **Примечания/комментарии** | Работник оформляет возврат книги. Книга удаляется из личного кабинета пользователя и ее статус изменяется на “доступный”. |

Таблица 11 – Тестовый пример 5

|  |  |
| --- | --- |
| **Тестовый пример #** | ТС\_ПИ\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Присвоение работником книги пользователю. |
| **Краткое изложение теста** | При вводе id книги и номера телефона пользователя и нажатии на кнопку “GIVE”, книга добавляется в личный кабинет пользователя |
| **Этапы теста** | 1. Ввод id книги и номера телефона пользователя, которому отдается книга. 2. Нажатие на кнопку “GIVE” |
| **Тестовые данные** | id книги: 10  номер телефона пользователя: 8(999)-999-99-99 |
| **Ожидаемый результат** | Книга добавляется в карточку читателя. |
| **Фактический результат** | Книга добавляется в карточку читателя. |
| **Статус** | Зачет |
| **Предварительное условие** | Книга является доступной, пользователь существует в базе данных. |
| **Постусловие** | Книга добавляется в карту пользователя. |
| **Примечания/комментарии** | Изначально у пользователя не было книги на руках, после ее бронирования работник присвоил книгу пользователю. Данные о книге перенеслись в личный кабинет пользователя. |

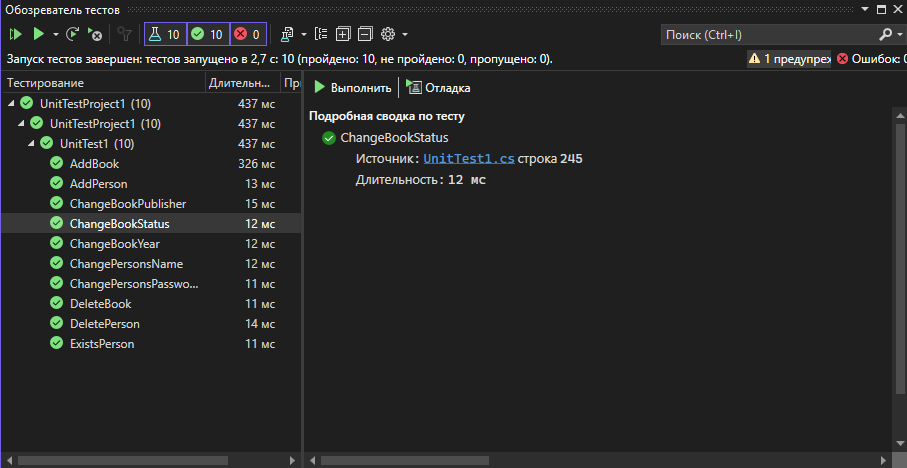


Рисунок 29. Успешное тестирование

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void AddPerson()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = " INSERT INTO Person([name], [phone\_number], [password], [role\_id]) Values(@name, @phone\_number, @password, 1)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@name", "Alex");

pr2 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

pr3 = new SqlParameter("@password", "jeNeS34");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.ExecuteNonQuery();

comDel = " SELECT Person.name FROM Person WHERE phone\_number = @phone\_number";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

var expected = "Alex";

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void ExistsPerson()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

try

{

string comDel = " INSERT INTO Person([name], [phone\_number], [password], [role\_id]) Values(@name, @phone\_number, @password, 1)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@name", "Alex");

pr2 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

pr3 = new SqlParameter("@password", "jeNeS34");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.ExecuteNonQuery();

dataBase.closeConnection();

}

catch

{

return;

}

Assert.Fail("Ожидаемое исключение не было получено");

}

[TestMethod]

public void ChangePersonsName()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "UPDATE Person Set name = @name WHERE phone\_number = @phone\_number";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@name", "Alexy");

pr2 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Person.name FROM Person WHERE phone\_number = @phone\_number";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

string expected = "Alexy";

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void ChangePersonsPassword()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "UPDATE Person Set password = @password WHERE phone\_number = @phone\_number";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@password", "89dJe34");

pr2 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Person.password FROM Person WHERE phone\_number = @phone\_number";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@phone\_number", "8(999)-999-99-99");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

string expected = "89dJe34";

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void DeletePerson()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "DELETE FROM Person WHERE phone\_number = @phone\_number";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1;

pr1 = new SqlParameter("@phone\_number", "8(932)-523-62-22");

cmd1.Parameters.Add(pr1);

cmd1.ExecuteNonQuery();

comDel = "SELECT Person.name FROM Person WHERE phone\_number = @phone\_number";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@phone\_number", "8(932)-523-62-22");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

dataBase.closeConnection();

Assert.AreEqual(null, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void AddBook()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = " INSERT INTO Book([name], [year], [publisher], [status\_id]) Values(@name, @year, @publisher, 1)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@name", "Erm");

pr2 = new SqlParameter("@year", "2004");

pr3 = new SqlParameter("@publisher", "Lyberts");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.ExecuteNonQuery();

comDel = " SELECT Book.year FROM Book WHERE name = @name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@name", "Erm");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

var expected = 2004;

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемый год не был получен!");

}

[TestMethod]

public void ChangeBookYear()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "UPDATE Book Set year = @year WHERE name = @name";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@year", 1981);

pr2 = new SqlParameter("@name", "1984");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Book.year FROM Book WHERE name = @name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@name", "1984");

cmd1.Parameters.Add(pr1);

int result = Convert.ToInt32(cmd1.ExecuteScalar());

int expected = 1981;

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void ChangeBookPublisher()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "UPDATE Book Set publisher = @publisher WHERE name = @name";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@publisher", "Morlos");

pr2 = new SqlParameter("@name", "1984");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Book.publisher FROM Book WHERE name = @name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@name", "1984");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

string expected = "Morlos";

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void ChangeBookStatus()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "UPDATE Book Set status\_id = @status\_id WHERE name = @name";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@status\_id", 2);

pr2 = new SqlParameter("@name", "Erm");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Book.status\_id FROM Book WHERE name = @name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@name", "Erm");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

int expected = 2;

dataBase.closeConnection();

Assert.AreEqual(expected, result, "Ожидаемое имя не было получено!");

}

[TestMethod]

public void DeleteBook()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string comDel = "DELETE FROM Book WHERE name = @name";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1;

pr1 = new SqlParameter("@name", "Erm");

cmd1.Parameters.Add(pr1);

cmd1.ExecuteNonQuery();

comDel = "SELECT Book.year FROM Book WHERE name = @name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@name", "2004");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

dataBase.closeConnection();

Assert.AreEqual(null, result, "Ожидаемое название не было получено!");

}

}

}