**1 Тестирование программного продукта**

* 1. **Обоснование методов и средств тестирования**

Разработанный программный продукт имеет небольшой функционал и, соответственно, небольшой объем кода, более того, приложение не выполняет никаких математических вычислений. По этой причине тестирование программными средствами, встроенными в Visual Studio 2019, не представляется возможным.

Для тестирования в программный код были добавлены блоки обработки исключений – специфический код, который срабатывает, когда программа аварийно завершает свою работу, хотя ошибок в коде никаких нет. В данном случае код выглядит следующим образом:

Form2.cs:

try

{

clientTableAdapter.Update(beauty\_salonDataSet);

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

return;

}

}

Form3.cs:

try

{

this.varnish\_coatingTableAdapter.Fill(this.beauty\_salonDataSet.varnish\_coating);

this.nail\_extensionsTableAdapter.Fill(this.beauty\_salonDataSet.nail\_extensions);

this.massageTableAdapter.Fill(this.beauty\_salonDataSet.massage);

this.pedicuresTableAdapter.Fill(this.beauty\_salonDataSet.pedicures);

this.manicureTableAdapter.Fill(this.beauty\_salonDataSet.manicure);

this.hairdressers\_womanTableAdapter.Fill(this.beauty\_salonDataSet.hairdressers\_woman);

this.hairdressers\_manTableAdapter.Fill(this.beauty\_salonDataSet.hairdressers\_man);

this.eyebrows\_and\_eyelashesTableAdapter.Fill(this.beauty\_salonDataSet.eyebrows\_and\_eyelashes);

this.clear\_faceTableAdapter.Fill(this.beauty\_salonDataSet.clear\_face);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных. Повторите попытку.");

}

За обработку исключений отвечают блоки try… catch.

1. **Тестирование**

**2.1 Тестовый граф и тестовые пути**

Чтобы последовательно и безошибочно протестировать программу, необходимо составить тестовый граф:

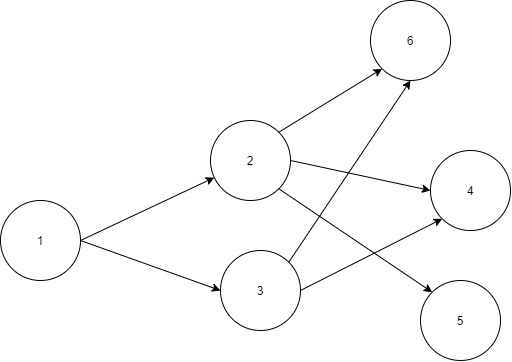


Рисунок 1 – Тестовый граф

Пояснения к графу:

1 открытие приложения

2 переход на вторую форму

3 переход на третью форму

4 возвращение на первую форму

5 запись клиента

6 выход

Для работы с тестовым графом необходимо составить тестовые пути, по которым будет тестироваться программа. Они будут выглядеть следующим образом:

1-2-4

1-3-4

1-2-6

1-3-6

1-2-5

**2.2 Интеграционное тестирование**

Далее по данным тестовым путям проводим интеграционное тестирование. **Интеграционное тестирование** – вид тестирования, которое подразумевает собой тестирование связей между различными модулями программы, их взаимодействие. Данные для интеграционного тестирования представлены ниже:

**ID** – 1

**Исходные данные:** открыта первая форма

**Цель:** тестирование перехода на вторую форму

**Описание:** пользователь, находясь на первой форме, нажимает на кнопку «Запись»

**Ожидаемый результат:** переход на вторую форму, если есть соединение с базой данных, если нет – сообщение об ошибке

**Результат:** открыта вторая форма (если отключить Open Server, тем самым разорвать соединение с сервером, будет выведено сообщение об ошибке соединения).

**ID** – 2

**Исходные данные:** открыта первая форма

**Цель:** протестировать переход на третью форму

**Описание:** пользователь, находясь на первой форме, нажимает на кнопку «Каталог»

**Ожидаемый результат:** результат: переход на третью форму, если есть соединение с базой данных, если нет – сообщение об ошибке

**Результат:** открыта третья форма (если отключить Open Server, тем самым разорвать соединение с сервером, будет выведено сообщение об ошибке соединения).

**ID** – 3

**Исходные данные:** открыта вторая форма

**Цель:** протестировать переход со второй формы на первую

**Описание:** пользователь, находясь на второй форме, нажимает на кнопку «Вернуться»

**Ожидаемый результат:** открывается первая форма, вторая форма скрывается

**Результат:** открыта первая форма, вторая форма исчезла

**ID** – 4

**Исходные данные:** открыта третья форма

**Цель:** протестировать переход с третьей формы на первую

**Описание:** пользователь, находясь на третьей форме, нажимает на кнопку «Вернуться»

**Ожидаемый результат:** открывается первая форма, третья форма скрывается

**Результат:** открыта первая форма, третья форма исчезла

**ID** – 5

**Исходные данные:** открыта вторая форма

**Цель:** протестировать корректное завершение работы приложения

**Описание:** находясь на второй форме, нажать кнопку «Выход»

**Ожидаемый результат:** закрытие приложения; приложение не остается в процессах

**Результат:** приложение закрывается и не остается в процессах

**ID** – 6

**Исходные данные:** открыта третья форма

**Цель:** протестировать корректное завершение работы

**Описание:** находясь на третьей форме, нажать на кнопку «Выход»

**Ожидаемый результат:** приложение закрывается и не остается в процессах

**Результат:** приложение закрыто, в процессах не остается

**ID** – 7

**Исходные данные:** открыта вторая форма

**Цель:** тестирование записи пользователя

**Описание:** находясь на второй форме, добавить личные данные в таблицу и нажать на кнопку «Добавить»

**Ожидаемый результат:** данные заносятся в базу данных

**Результат:** данные занесены в базу данных

Скриншоты тестирования (ID - 7)

Изначально в базе имелись такие данные:

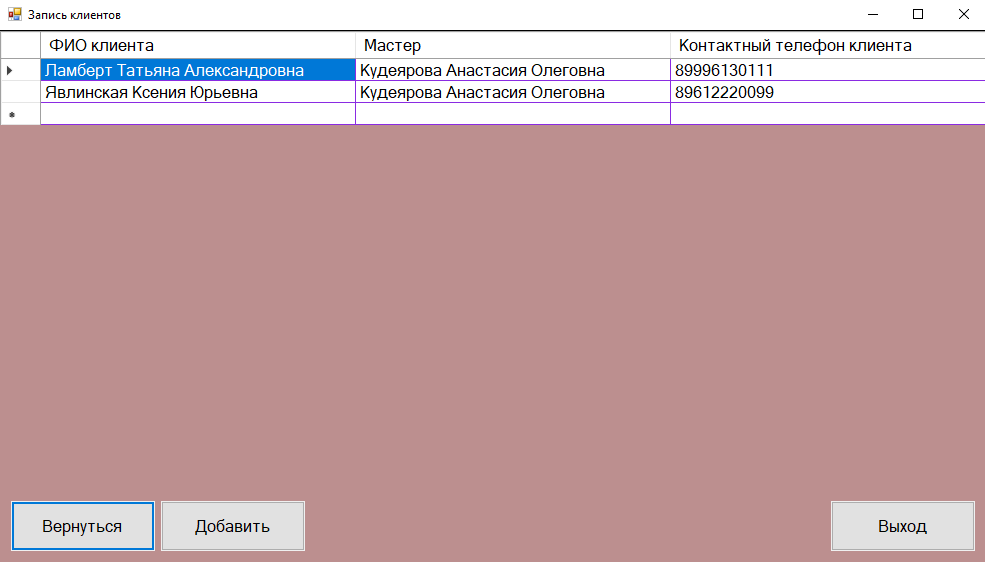


Рисунок 2 – Вторая форма

Далее вписываем в таблицу тестовые данные и нажимаем кнопку «Добавить»:

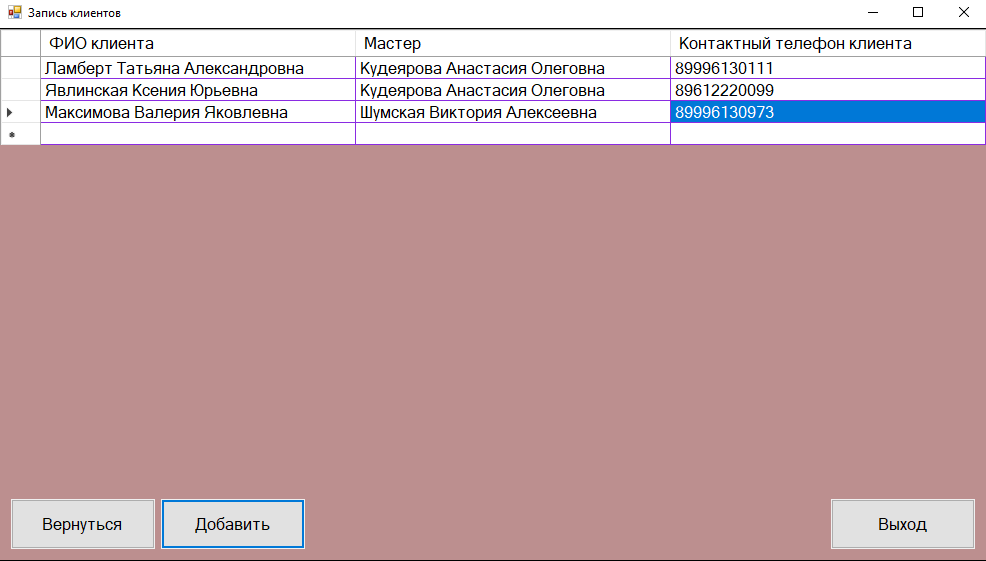


Рисунок 3 – Занесение данных

Проверяем наличие записи в базе данных:

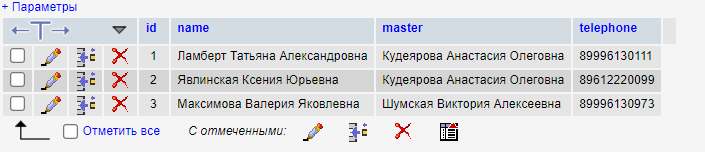


Рисунок 4 – Запись добавлена в базу данных

**3 Результаты тестирования**

Система прошла все этапы тестирования и успешно показала свою работоспособность. Багов и недочетов обнаружено не было.

Интеграционное тестирование производилось с помощью тестового графа и тестовых путей. Все тестовые пути были пройдены успешно.

Модульное тестирование производилось с помощью try-catch блоков. Подобные блоки были добавлены во все модули кода, что поможет в будущем избежать критических ситуаций в работе программы.

Из этого следует, что программный продукт полностью готов к работе в реальных условиях.