Тестируемая система: Order Manager v2.0

1. Идентификация

| Параметр | Значение |
| --- | --- |
| Название проекта | OrderManager |
| Версия ПО | 2.0 |
| Дата тестирования | 23.10.25-28.10.25 |
| Тип тестирования | Функциональное, UI-тестирование |
| Тест-лид | Кликин Егор Алексеевич |

1. Сводка тестирования:

Цели:

* Проверить корректность работы GUI приложения
* Убедиться в правильности:
  + Добавления, удаления заказов
  + Обновления статуса заказа
  + Уведомления пользователя об изменении статуса заказа
  + Возможность отключения уведомления пользователя об изменении определенных статусов заказов
  + Хранение данных и загрузка из файла

Критерии входа:

* Наличие тестовых сценариев
* Готовность тестовой среды

Критерии выхода:

* 95% успешных тест-кейсов
* Отсутствие критических дефектов

Методы тестирования:

* Unit тесты
* Ручное функциональное тестирования (GUI)
* Автоматизированное функциональное тестирование (GUI)

1. Охват тестирования

Unit тесты:

Класс Order: 100% покрытие тестами

Всего тестов: 2

Класс OrderManager: 100% покрытие тестами

Всего тестов: 7

Класс NotificationManager: 100% покрытие тестами

Всего тестов: 8

Автоматическое UI-тестирование

75% покрытие тестами

Всего тестов: 12

| Компонент | Тест-кейсы | Охват (%) |
| --- | --- | --- |
| Создание заказа | 3 | 100% |
| Удаление заказа | 2 | 100% |
| Обновление статуса заказа | 3 | 100% |
| Уведомления пользователя об изменении статуса заказа | 3 | 100% |
| Возможность отключения уведомления пользователя об изменении определенных статусов заказов | 1 | 100% |
| Хранение данных и загрузка из файла | 0 | 0% |

Ручное UI-тестирование

100% покрытие тестами

Всего тестов: 16

| Компонент | Тест-кейсы | Охват (%) |
| --- | --- | --- |
| Создание заказа | 3 | 100% |
| Удаление заказа | 2 | 100% |
| Обновление статуса заказа | 3 | 100% |
| Уведомления пользователя об изменении статуса заказа | 3 | 100% |
| Возможность отключения уведомления пользователя об изменении определенных статусов заказов | 1 | 100% |
| Хранение данных и загрузка из файла | 4 | 100% |

1. Метрики и результаты Unit-тесты

| Название | Описание | Результат |
| --- | --- | --- |
| OrderConstuctor\_Test1 | Создание нового заказа | Passed |
| UpdateStatus\_Test1 | Обновление статуса заказа | Passed |
| OrderManagerConstructor\_Test1 | Создание нового менеджера заказов | Passed |
| OrderManagerAddNullOrder\_Test1 | Добавление null-заказа в менеджер заказов | Passed |
| OrderManagerAddOrder\_Test2 | Успешное добавление заказа в менеджер заказов | Passed |
| OrderManagerRemoveNullOrder\_Test1 | Удаление null-заказа из менеджер заказов | Passed |
| OrderManagerRemoveOrder\_Test2 | Успешное добавление заказа из менеджера заказов | Passed |
| OrderManagerUpdateNullOrder\_Test1 | Обновление статуса null-заказа в менеджере заказов | Passed |
| OrderManagerUpdateOrder\_Test2 | Успешное обновление статуса заказа из менеджера заказов | Passed |
| NotificationManagerConstructor\_Test1 | Создание нового менеджера уведомлений | Passed |
| NotificationManagerChangeStatusOnInProgressNotifyOn\_Test | Создание сообщения для уведомления при изменении статуса на “В обработке”, когда уведомления для взятия заказа в обработку включены | Passed |
| NotificationManagerChangeStatusOnCompletedNotifyOn\_Test | Создание сообщения для уведомления при изменении статуса на “Завершен”, когда уведомления для завершения заказа включены | Passed |
| NotificationManagerChangeStatusOnInProgressNotifyOff\_Test | Создание сообщения для уведомления при изменении статуса на “В обработке”, когда уведомления для взятия заказа в обработку выключены | Passed |
| NotificationManagerChangeStatusOnCompletedNotifyOff\_Test | Создание сообщения для уведомления при изменении статуса на “Завершен”, когда уведомления для завершения заказа выключены | Passed |
| NotificationManagerChangeNotifyOnInProgress\_Test | Изменение свойства, отвечающее за включение и выключение уведомлений для взятия заказа в обработку | Passed |
| NotificationManagerChangeNotifyOnCompleted\_Test | Изменение свойства, отвечающее за включение и выключение уведомлений для завершения заказа | Passed |
| NotificationManagerChangeOnTheSameStatus\_Test | Возвращение пустого сообщения при обновлении статуса на тот же самый | Passed |

Автоматизированное UI - тестирование

| Категория тестов | Всего | Passed | Failed | Охват (%) |
| --- | --- | --- | --- | --- |
| Создание заказа | 3 | 3 | 0 | 100% |
| Удаление заказа | 2 | 2 | 0 | 100% |
| Обновление статуса заказа | 3 | 3 | 0 | 100% |
| Уведомления пользователя об изменении статуса заказа | 3 | 3 | 0 | 100% |
| Возможность отключения уведомления пользователя об изменении определенных статусов заказов | 1 | 1 | 0 | 100% |
| Хранение данных и загрузка из файла | 0 | 0 | 0 | - |

Ручное UI - тестирование

| Категория тестов | Всего | Passed | Failed | Охват (%) |
| --- | --- | --- | --- | --- |
| Создание заказа | 3 | 3 | 0 | 100% |
| Удаление заказа | 2 | 2 | 0 | 100% |
| Обновление статуса заказа | 3 | 3 | 0 | 100% |
| Уведомления пользователя об изменении статуса заказа | 3 | 3 | 0 | 100% |
| Возможность отключения уведомления пользователя об изменении определенных статусов заказов | 1 | 1 | 0 | 100% |
| Хранение данных и загрузка из файла | 4 | 4 | 0 | 100% |

1. Оценка дефектов

На текущий момент критические дефекты не обнаружены.

1. Выводы и рекомендации

Выводы:

● Все основные функции работают корректно;

● 100% тестов пройдено успешно.

Рекомендации:

1. Добавить:

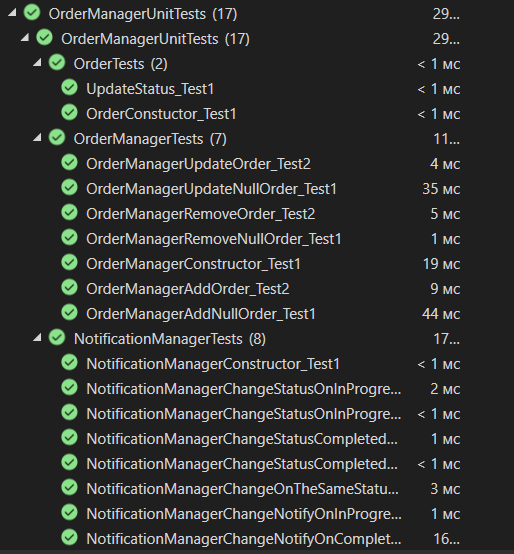
● Подтверждение при удалении заказа;

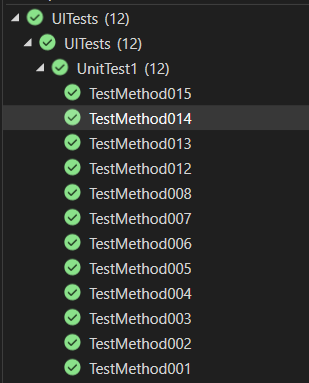
● Подтверждение при обновлении статуса заказа;

Заключение: Приложение соответствует основным требованиям и готово к релизу после исправления незначительных UI-проблем. Автоматическое UI тестирование не покрывает все возможные тесты, которые возможно провести вручную.

Приложение А

Скриншот выполнения Unit и UI тестов:





Приложение Б

Код тестирования:

[OrderTests.cs](http://ordertests.cs):

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using order\_controller;

namespace OrderModuleTest

{

[TestClass]

public class OrderTests

{

[TestMethod]

public void OrderConstuctor\_Test1()

{

string name = "Василий";

string description = "2 банки колы";

DateTime dateTime = DateTime.Now;

Order order = new Order(name, description, dateTime);

Assert.IsNotNull(order);

Assert.AreEqual(name, order.CustomerName);

Assert.AreEqual(description, order.Description);

Assert.AreEqual(dateTime, order.CreationDate);

Assert.AreEqual(OrderStatus.Новый, order.Status);

}

[TestMethod]

public void UpdateStatus\_Test1()

{

OrderStatus newStatus = OrderStatus.Завершён;

Order order = new Order("order", "description", DateTime.Now);

order.UpdateStatus(newStatus);

Assert.AreEqual(newStatus, order.Status);

}

}

}

[OrderManagerTests.cs](http://ordermanagertests.cs)

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using order\_controller;

namespace OrderModuleTest

{

[TestClass]

public class OrderManagerTests

{

[TestMethod]

public void OrderManagerConstructor\_Test1() // тест для конструктора

{

OrderManager manager = new OrderManager();

Assert.IsNotNull(manager);

}

[TestMethod]

[ExpectedException(typeof(ArgumentNullException))]

public void OrderManagerAddNullOrder\_Test1() // тест для проверки выкидывания исключения при попытки добавления null-заказа

{

OrderManager manager = new OrderManager();

manager.AddOrder(null);

}

[TestMethod]

public void OrderManagerAddOrder\_Test2() // тест для проверки успешного добавления заказа

{

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderManager manager = new OrderManager();

Order order = new Order(name, description, dateTime);

manager.AddOrder(order);

Assert.IsNotNull(manager.Orders.Find(x => x.CreationDate == dateTime));

Assert.AreEqual(name, manager.Orders.Find(x => x.CreationDate == dateTime).CustomerName);

Assert.AreEqual(description, manager.Orders.Find(x => x.CreationDate == dateTime).Description);

Assert.AreEqual(dateTime, manager.Orders.Find(x => x.CreationDate == dateTime).CreationDate);

}

[TestMethod]

[ExpectedException(typeof(ArgumentNullException))]

public void OrderManagerRemoveNullOrder\_Test1() // тест для проверки выкидывания исключения при попытки удаления null-заказа

{

OrderManager manager = new OrderManager();

manager.RemoveOrder(null);

}

[TestMethod]

public void OrderManagerRemoveOrder\_Test2() // тест для проверки успешного удаления заказа

{

string name1 = "Роман";

string description1 = "2 пачки кофе";

DateTime dateTime1 = new DateTime(2025, 10, 12);

string name2 = "Марк";

string description2 = "2 пачки конфет";

DateTime dateTime2 = new DateTime(2022, 12, 12);

Order order1 = new Order(name1, description1, dateTime1);

Order order2 = new Order(name2, description2, dateTime2);

OrderManager manager = new OrderManager();

manager.AddOrder(order1);

manager.AddOrder(order2);

manager.RemoveOrder(order1);

Assert.AreEqual(-1, manager.Orders.FindIndex(x => x.CreationDate == dateTime1));

Assert.AreEqual(-1, manager.Orders.FindIndex(x => x.CreationDate == dateTime1));

}

[TestMethod]

[ExpectedException(typeof(ArgumentNullException))]

public void OrderManagerUpdateNullOrder\_Test1() // тест для проверки выкидывания исключения при попытки обноления статуса у null-заказа

{

OrderManager manager = new OrderManager();

manager.UpdateOrderStatus(null, OrderStatus.Завершён);

}

[TestMethod]

public void OrderManagerUpdateOrder\_Test2() // тест для проверки успешного обновления статуса заказа

{

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderManager manager = new OrderManager();

Order order = new Order(name, description, dateTime);

OrderStatus newStatus = OrderStatus.В\_обработке;

manager.AddOrder(order);

manager.UpdateOrderStatus(order, newStatus);

Assert.AreEqual(newStatus, manager.Orders.Find(x => x.CreationDate == dateTime).Status);

}

}

}

[NotificationManagerTests.cs](http://notificationmanagertests.cs)

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using order\_controller;

namespace OrderModuleTest

{

[TestClass]

public class NotificationManagerTests

{

[TestMethod]

public void NotificationManagerConstructor\_Test1() // тест для конструктора без параметров

{

NotificationManager manager = new NotificationManager();

Assert.IsNotNull(manager);

}

[TestMethod]

public void NotificationManagerChangeStatusOnInProgressNotifyOn\_Test() // тест на формирование сообщения, когда уведомления для взятия в обработку включены

{

NotificationManager manager = new NotificationManager();

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderStatus newStatus = OrderStatus.В\_обработке;

Order order = new Order(name, description, dateTime);

order.UpdateStatus(newStatus);

string message = manager.NotifyStatusChange(order, OrderStatus.Новый, newStatus);

Assert.IsNotNull(message);

Assert.IsTrue(message.Contains(name));

Assert.IsTrue(message.Contains(description));

Assert.IsTrue(message.Contains(dateTime.ToString()));

Assert.IsTrue(message.Contains("взят в обработку"));

}

[TestMethod]

public void NotificationManagerChangeStatusCompletedNotifyOn\_Test() // тест на формирование сообщения, когда уведомления для завершения заказа включены

{

NotificationManager manager = new NotificationManager();

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderStatus newStatus = OrderStatus.Завершён;

Order order = new Order(name, description, dateTime);

order.UpdateStatus(newStatus);

string message = manager.NotifyStatusChange(order, OrderStatus.Новый, newStatus);

Assert.IsNotNull(message);

Assert.IsTrue(message.Contains(name));

Assert.IsTrue(message.Contains(description));

Assert.IsTrue(message.Contains(dateTime.ToString()));

Assert.IsTrue(message.Contains("завершен"));

}

[TestMethod]

public void NotificationManagerChangeStatusOnInProgressNotifyOff\_Test() // тест на формирование сообщения, когда уведомления для взятия в обработку выключены

{

NotificationManager manager = new NotificationManager();

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderStatus newStatus = OrderStatus.В\_обработке;

Order order = new Order(name, description, dateTime);

manager.NotifyOnInProgress = false;

order.UpdateStatus(newStatus);

string message = manager.NotifyStatusChange(order, OrderStatus.Новый, newStatus);

Assert.IsNotNull(message);

Assert.AreEqual(message, "");

}

[TestMethod]

public void NotificationManagerChangeStatusCompletedNotifyOff\_Test() // тест на формирование сообщения, когда уведомления для завершения заказа выключены

{

NotificationManager manager = new NotificationManager();

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderStatus newStatus = OrderStatus.Завершён;

Order order = new Order(name, description, dateTime);

manager.NotifyOnCompleted = false;

order.UpdateStatus(newStatus);

string message = manager.NotifyStatusChange(order, OrderStatus.Новый, newStatus);

Assert.IsNotNull(message);

Assert.AreEqual(message, "");

}

[TestMethod]

public void NotificationManagerChangeNotifyOnInProgress\_Test() // тест для свойства notifyOnInProgress, отвечающее за включение и выключение уведомлений для взятия закказа в обработку

{

NotificationManager manager = new NotificationManager();

manager.NotifyOnInProgress = false;

Assert.IsFalse(manager.NotifyOnInProgress);

}

[TestMethod]

public void NotificationManagerChangeNotifyOnCompleted\_Test() // тест для свойства notifyOnInProgress, отвечающее за включение и выключение уведомлений для завершения заказа

{

NotificationManager manager = new NotificationManager();

manager.NotifyOnCompleted = false;

Assert.IsFalse(manager.NotifyOnCompleted);

}

[TestMethod]

public void NotificationManagerChangeOnTheSameStatus\_Test() // тест для проверки возвращения пустой строки при обновлении статуса на тот же самый

{

NotificationManager manager = new NotificationManager();

string name = "Роман";

string description = "2 пачки кофе";

DateTime dateTime = DateTime.Now;

OrderStatus newStatus = OrderStatus.Завершён;

Order order = new Order(name, description, dateTime);

order.UpdateStatus(newStatus);

order.UpdateStatus(newStatus);

string message = manager.NotifyStatusChange(order, newStatus, newStatus);

Assert.IsNotNull(message);

Assert.AreEqual(message, "");

}

}

}

[UnitTest1.cs](http://unittest1.cs)

using FlaUI.UIA3;

using FlaUI.Core;

using FlaUI.Core.AutomationElements;

using Microsoft.VisualStudio.TestPlatform.CommunicationUtilities;

namespace AutoTest

{

[TestClass]

public class UnitTest1

{

private Application? \_app;

private UIA3Automation? \_automation;

private Window? \_mainWindow;

private string filePath = @"D:\даша\учеба колледж\3курс\тестирование\5 лаба\lab1\_pirat\_beb\NoteForm\NoteForm\bin\Debug\notes.txt";

[TestInitialize]

public void TestInitialize()

{

\_app = Application.Launch(@"D:\даша\учеба колледж\3курс\тестирование\5 лаба\lab1\_pirat\_beb\NoteForm\NoteForm\bin\Debug\NoteForm.exe");

\_automation = new UIA3Automation();

\_mainWindow = \_app.GetMainWindow(\_automation);

}

[TestCleanup]

public void TestCleanup()

{

\_automation?.Dispose();

\_app?.Close();

if (File.Exists(filePath))

{

File.WriteAllText(filePath, string.Empty);

}

}

private void CreateNote(string title, string content, params string[] tags)

{

var nameTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("titleTextBox"))?.AsTextBox();

var contentTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("contentTextBox"))?.AsTextBox();

var addNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addNoteButton"))?.AsButton();

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

foreach (var checkBox in tagsCheckedListBox)

{

foreach (var tag in tags)

{

if (checkBox.AsLabel().Text == tag)

{

checkBox.Click();

checkBox.Click();

}

}

}

nameTextBox?.Enter(title);

contentTextBox?.Enter(content);

addNoteButton?.Click();

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

}

private void CreateNote(string title, string content)

{

var nameTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("titleTextBox"))?.AsTextBox();

var contentTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("contentTextBox"))?.AsTextBox();

var addNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addNoteButton"))?.AsButton();

nameTextBox?.Enter(title);

contentTextBox?.Enter(content);

addNoteButton?.Click();

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

}

private void CreateTag(string title)

{

var newTagTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("newTagTextBox"))?.AsTextBox();

var addTagButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addTagButton"))?.AsButton();

newTagTextBox?.Enter(title);

addTagButton?.Click();

}

[TestMethod]

public void TestMethod1()

{

var title = "Список покупок";

var content = "Хлеб, колбаса";

CreateNote(title, content);

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lastNote = notesList?.Items.Last()?.Text;

Assert.IsTrue(lastNote?.Contains(title));

Assert.IsTrue(lastNote?.Contains(content));

TestCleanup();

}

[TestMethod]

public void TestMethod2()

{

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtBefore = notesList?.Items.Count();

var nameTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("titleTextBox"))?.AsTextBox();

var addNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addNoteButton"))?.AsButton();

nameTextBox?.Enter("Отжуманя");

addNoteButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

var lengthAfter = notesList?.Items.Count();

Assert.AreEqual("Заполните все поля!", message);

Assert.AreEqual(lenghtBefore, lengthAfter);

TestCleanup();

}

[TestMethod]

public void TestMethod3()

{

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtBefore = notesList?.Items.Count();

var nameTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("titleTextBox"))?.AsTextBox();

var addNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addNoteButton"))?.AsButton();

nameTextBox?.Enter("Список дел:");

addNoteButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

var lengthAfter = notesList?.Items.Count();

Assert.AreEqual("Заполните все поля!", message);

Assert.AreEqual(lenghtBefore, lengthAfter);

TestCleanup();

}

[TestMethod]

public void TestMethod4()

{

var title = "Компьютер";

var content = "Процессор";

CreateNote(title, content);

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtBefore = notesList?.Items.Count();

var deleteNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("removeNoteButton"))?.AsButton();

notesList?.Select(0);

deleteNoteButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

var lengthAfter = notesList?.Items.Count();

Assert.AreEqual(lenghtBefore - 1, lengthAfter);

TestCleanup();

}

[TestMethod]

public void TestMethod5()

{

var notesList = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtBefore = notesList?.Items.Count();

var deleteNoteButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("removeNoteButton"))?.AsButton();

deleteNoteButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

var lengthAfter = notesList?.Items.Count();

Assert.AreEqual(lenghtBefore, lengthAfter);

Assert.AreEqual("Выберите заметку для удаления!", message);

TestCleanup();

}

[TestMethod]

public void TestMethod10()

{

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lenghtCheckedBoxBefore = tagsCheckedListBox?.Length;

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox"))?.AsListBox();

var lenghtListBoxBefore = allTagsListBox?.Items.Count();

var title = "Покупки";

CreateTag(title);

tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lengthCheckedBoxAfter = tagsCheckedListBox?.Length;

var lengthListBoxAfter = allTagsListBox?.Items.Length;

Assert.AreEqual(lenghtCheckedBoxBefore + 1, lengthCheckedBoxAfter);

Assert.AreEqual(lenghtListBoxBefore + 1, lengthListBoxAfter);

string[] lines = File.ReadAllLines(filePath);

string? firstLine = lines.LastOrDefault();

Assert.IsTrue(firstLine?.Contains($"{title}"));

TestCleanup();

}

[TestMethod]

public void TestMethod11()

{

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lenghtCheckedBoxBefore = tagsCheckedListBox?.Length;

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox"))?.AsListBox();

var lenghtListBoxBefore = allTagsListBox?.Items.Count();

var addTagButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("addTagButton"))?.AsButton();

addTagButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

okButton?.Click();

var lengthCheckedBoxAfter = tagsCheckedListBox?.Length;

var lengthListBoxAfter = allTagsListBox?.Items.Count();

Assert.AreEqual(lenghtCheckedBoxBefore, lengthCheckedBoxAfter);

Assert.AreEqual(lenghtListBoxBefore, lengthListBoxAfter);

Assert.IsTrue(message.Contains("Пустой тег"));

TestCleanup();

}

[TestMethod]

public void TestMethod12()

{

var title = "Покупки";

CreateTag(title);

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lenghtCheckedBoxBefore = tagsCheckedListBox?.Length;

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox"))?.AsListBox();

var lenghtListBoxBefore = allTagsListBox?.Items.Count();

CreateTag(title);

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

var lengthCheckedBoxAfter = tagsCheckedListBox?.Length;

var lengthListBoxAfter = allTagsListBox?.Items.Count();

Assert.AreEqual(lenghtCheckedBoxBefore, lengthCheckedBoxAfter);

Assert.AreEqual(lenghtListBoxBefore, lengthListBoxAfter);

Assert.IsTrue(message.Contains("тег уже существует"));

TestCleanup();

}

[TestMethod]

public void TestMethod13()

{

var title = "Ноутбук";

CreateTag(title);

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lenghtCheckedBoxBefore = tagsCheckedListBox?.Length;

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox"))?.AsListBox();

var lenghtListBoxBefore = allTagsListBox?.Items.Count();

var removeTagButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("removeTagButton"))?.AsButton();

allTagsListBox?.Select(0);

removeTagButton?.Click();

tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lengthCheckedBoxAfter = tagsCheckedListBox?.Length;

var lengthListBoxAfter = allTagsListBox?.Items.Count();

Assert.AreEqual(lenghtCheckedBoxBefore - 1, lengthCheckedBoxAfter);

Assert.AreEqual(lenghtListBoxBefore - 1, lengthListBoxAfter);

TestCleanup();

}

[TestMethod]

public void TestMethod14()

{

var title = "заголовок";

CreateTag(title);

var tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lenghtCheckedBoxBefore = tagsCheckedListBox?.Length;

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox"))?.AsListBox();

var lenghtListBoxBefore = allTagsListBox?.Items.Count();

var removeTagButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("removeTagButton"))?.AsButton();

removeTagButton?.Click();

var messageBox = \_mainWindow?.ModalWindows.FirstOrDefault();

var messageText = messageBox?.FindFirstDescendant(cf => cf.ByAutomationId("65535"));

string message = messageText?.Name;

var okButton = messageBox?.FindFirstDescendant(x => x.ByAutomationId("2")).AsButton();

okButton?.Click();

tagsCheckedListBox = \_mainWindow?.FindAllDescendants(x => x.ByControlType(FlaUI.Core.Definitions.ControlType.CheckBox));

var lengthCheckedBoxAfter = tagsCheckedListBox?.Length;

var lengthListBoxAfter = allTagsListBox?.Items.Count();

Assert.AreEqual(lenghtCheckedBoxBefore, lengthCheckedBoxAfter);

Assert.AreEqual(lenghtListBoxBefore, lengthListBoxAfter);

Assert.IsTrue(message.Contains("Выберите тег"));

TestCleanup();

}

[TestMethod]

public void TestMethod15()

{

var tag = "Списки";

var title = "Список дел";

var content = "Покушать";

CreateTag(tag);

var listNotes = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtListNotesBefore = listNotes?.Items.Count();

CreateNote(title, content, tag);

var lenghtListNotesAfter = listNotes?.Items.Count();

Assert.AreEqual(lenghtListNotesBefore + 1, lenghtListNotesAfter);

var lastItem = listNotes?.Items.LastOrDefault();

Assert.IsTrue(lastItem?.Text.Contains($"{tag}"));

string[] lines = File.ReadAllLines(filePath);

string? lastLine = lines.LastOrDefault();

Assert.IsTrue(lastLine?.StartsWith($"{title}|{content}"), "Неверное содержание заметки в файле.");

Assert.IsTrue(lastLine?.Contains($"|{tag}"), $"Ожидался тег {tag} в строке, но получено: {lastLine}");

TestCleanup();

}

[TestMethod]

public void TestMethod16()

{

var tag1 = "Списки";

var tag2 = "дом";

var title = "Список дел";

var content = "Покушать";

CreateTag(tag1);

CreateTag(tag2);

var listNotes = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

var lenghtListNotesBefore = listNotes?.Items.Count();

CreateNote(title, content, tag1, tag2);

var lenghtListNotesAfter = listNotes?.Items.Count();

Assert.AreEqual(lenghtListNotesBefore + 1, lenghtListNotesAfter);

var lastItem = listNotes?.Items.LastOrDefault();

Assert.IsTrue(lastItem?.Text.Contains($"{tag1}, {tag2}"));

string[] lines = File.ReadAllLines(filePath);

string? lastLine = lines.LastOrDefault();

Assert.IsTrue(lastLine?.StartsWith($"{title}|{content}"), "Неверное содержание заметки в файле.");

Assert.IsTrue(lastLine?.Contains($"|{tag1},{tag2}"), $"Ожидался тег {tag1} и {tag2} в строке, но получено: {lastLine}");

TestCleanup();

}

[TestMethod]

public void TestMethod17()

{

var tag1 = "Списки";

var tag2 = "дом";

var title = "Список дел";

var content = "Покушать";

var title1 = "Работа";

var content1 = "Лабораторная работа";

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox")).AsListBox();

CreateTag(tag1);

CreateTag(tag2);

CreateNote(title, content, tag1);

CreateNote(title, content, tag2);

CreateNote(title1, content1, tag1);

CreateNote(title1, content1, tag2);

allTagsListBox?.Focus();

allTagsListBox?.Items[0].Click();

var listNotes = \_mainWindow?.FindFirstDescendant(x => x.ByName("notesListBox"))?.AsListBox();

int i = 0;

foreach (var note in listNotes?.Items)

{

if (!note.Text.Contains(tag1)) i++;

}

Assert.AreEqual(0, i);

TestCleanup();

}

[TestMethod]

public void TestMethod18()

{

var tag = "Списки";

var newTag = "Уроки";

var title = "Список дел";

var content = "Покушать";

var allTagsListBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("allTagsListBox")).AsListBox();

var editTagButton = \_mainWindow?.FindFirstDescendant(x => x.ByName("editTagButton")).AsButton();

var editTagTextBox = \_mainWindow?.FindFirstDescendant(x => x.ByName("editTagTextBox")).AsTextBox();

CreateTag(tag);

CreateNote(title, content, tag);

allTagsListBox?.Focus();

allTagsListBox?.Items[0].Click();

editTagTextBox?.Enter(newTag);

editTagButton?.Click();

var item = allTagsListBox?.Items.LastOrDefault();

Assert.IsTrue(item?.Text.Contains(newTag));

}

}

}

Приложение В

Список всех тест-кейсов:

1. Добавление заказа с корректными данными
2. Добавление заказа с пустым именем клиента
3. Добавление заказа с пустым описанием
4. Удаление существующего заказа
5. Удаление заметки без выбора
6. Изменение статуса заказа на "В обработке"
7. Изменение статуса заказа без выбора статуса заказа
8. Изменение статуса заказа без выбора статуса заказа
9. Загрузка списка при несуществовании файла для записи
10. Загрузка списка, где в файле notes.txt одна запись должна быть повреждена
11. Наличие хотя бы одной записи в файле orders.txt
12. Сохранение данных в формате UTF-8
13. Уведомление при изменении статуса заказа на "Завершён"
14. Уведомление при изменении статуса заказа на "В обработке"
15. Отсутствие уведомления при отключенной настройке статуса
16. Уведомления не появляются при том же статусе

Приложение Г

Тест-кейсы:

1. **Добавление заказа с корректными данными**

* **Идентификатор:** TC-001
* **Название:** Добавление заказа с корректными данными
* **Предусловия:** Нет
* **Шаги выполнения:**

1. Ввести в поле “Имя клиента” значение: “Анатолий”
2. Ввести в поле “Описание” значение: “заказ 2”
3. Выбрать в поле “Дата создания заказа” значение: 15 октября 2025
4. Нажать на кнопку “Добавить”

* **Ожидаемый результат:** Заказ успешно добавлен, отображается в списке заказов, в файле заказ должен отображаться в виде: “Анатолий|заказ 2|0|Анатолий|заказ 2|0|2025-10-15 13 :56:53
* **Фактический результат:** Заказ успешно добавлен, отображается в списке заказов, в файле заказ должен отображаться в виде: “Анатолий|заказ 2|0|Анатолий|заказ 2|0|2025-10-15 13 :56:53
* **Статус:** Passed

1. **Добавление заказа с пустым именем клиента**

* **Идентификатор: TC-002**
* **Название:** Добавление заказа с пустым именем клиента
* **Предусловия:** Нет
* **Шаги выполнения:**

1. Оставить поле "Имя клиента" пустым
2. Ввести в поле "Описание" значение: "Срочный заказ"
3. Установить текущую дату в поле "Дата создания заказа"
4. Нажать кнопку "Добавить"

* **Ожидаемый результат:** Появится сообщение об ошибке: “Заполните все поля!”, запись не добавлена в файл
* **Фактический результат:** Появится сообщение об ошибке: “Заполните все поля!”, запись не добавлена в файл
* **Статус:** Passed

1. **Добавление заказа с пустым описанием**

* **Идентификатор:** TC-003
* **Название:** Добавление заказа с пустым описанием
* **Предусловия:** Нет
* **Шаги выполнения:**

1. Ввести в поле “Имя клиента” значение: “Анатолий”
2. Оставить поле “Описание” пустым
3. Установить текущую дату в поле "Дата создания заказа"
4. Нажать кнопку "Добавить"

* **Ожидаемый результат:** Появится сообщение об ошибке: “Заполните все поля!”, запись не добавлена в файл
* **Фактический результат:** Появится сообщение об ошибке: “Заполните все поля!”, запись не добавлена в файл
* **Статус:** Passed

1. **Удаление существующего заказа**

* **Идентификатор:** TC-004
* **Название:** Удаление существующего заказа
* **Предусловия:** Наличие хотя бы одного заказа в списке
* **Шаги выполнения:**

1. Нажать на первый заказ
2. Нажать кнопку "Удалить"

* **Ожидаемый результат:** Заказ удален из файла orders.txt, он не отображается в списке заказов
* **Фактический результат:** Заказ не удален из файла orders.txt, он отображается в списке заказов  
  **Статус:** Failed

1. **Удаление заметки без выбора**

* **Идентификатор:** TC-005
* **Название:** Удаление заметки без выбора
* **Предусловия:** Нет
* **Шаги выполнения:**

1. Нажать на кнопку “Удалить”

* **Ожидаемый результат:** Отобразиться сообщение об ошибке: “Выберите заказ для удаления”
* **Фактический результат:** Отобразиться сообщение об ошибке: “Выберите заказ для удаления”
* **Статус:** Passed

1. **Изменение статуса заказа на “В обработке”**

* **Идентификатор:** TC-006
* **Название:** Изменение статуса заказа на "В обработке"
* **Предусловия:** В системе есть заказ со статусом "Новый"
* **Шаги выполнения:**

1. Выбрать первый заказ в списке со статусом “Новый”
2. Изменить статус на "В обработке" через выпадающий список “Статус”
3. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Статус заказа обновлен в списке (“В обработке”) и в файле (значение 1)
* **Фактический результат:** Статус заказа не изменился в списке (“Новый”) и в файле (значение 0)  
  **Статус:** Failed

1. **Изменение статуса заказа без выбора статуса заказа**

* **Идентификатор:** TC-007
* **Название:** Изменение статуса заказа без выбора статуса заказа
* **Предусловия:** В системе есть заказ хотя бы один заказ
* **Шаги выполнения:**

1. Выбрать заказ в списке
2. Не выбирать статус заказа через выпадающий список “Статус”
3. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Отобразиться сообщение об ошибке: “Выберите статус для обновления статуса заказа”, заказ не изменился в списке заказов и в файле order.txt
* **Фактический результат:** Сообщение об ошибке не отобразилось, заказ не изменился в списке заказов и в файле order.txt  
  **Статус:** Failed

1. **Изменение статуса заказа без выбора заказа**

**Идентификатор:** TC-008

**Название:** Изменение статуса заказа без выбора статуса заказа

**Предусловия:** В системе есть заказ хотя бы один заказ

**Шаги выполнения:**

Не выбирать заказ в списке заказов

Выбрать статус “В обработке” через выпадающий список “Статус”

Нажать на кнопку “Обновить статус”

**Ожидаемый результат:** Отобразиться сообщение об ошибке: “Выберите заказ для обновления статуса!”

**Фактический результат:** Отобразиться сообщение об ошибке: “Выберите заказ для обновления статуса!”

**Статус:** Passed

1. **Загрузка списка при несуществовании файла для записи**

* **Идентификатор:** TC-009
* **Название:** Загрузка списка при несуществовании файла для записи
* **Предусловия:** Если order.txt существует, то удалить (order\_controller/bin/Debug/order.txt)
* **Шаги выполнения:**

1. Запустить программу

* **Ожидаемый результат:** Вывод пустого списка
* **Фактический результат:** Вывод пустого списка
* **Статус:** Passed

1. **Загрузка списка, где в файле notes.txt одна запись должна быть повреждена**

* **Идентификатор:** TC-010
* **Название:** Загрузка списка, где в файле notes.txt одна запись должна быть повреждена
* **Предусловия:** Наличие хотя бы одной записи в файле notes.txt
* **Шаги выполнения:**

1. Удалить один символ "|" из последней строки в файле notes.txt
2. Запустить программу

* **Ожидаемый результат:** Список загружен и отображается, пропуск поврежденной записи
* **Фактический результат:** Список загружен и отображается, пропуск поврежденной записи
* **Статус:** Passed

1. **Загрузка заказов из файла**

* **Идентификатор:** TC-011
* **Название:** Загрузка заказов из файла
* **Предусловия:** Наличие хотя бы одной записи в файле orders.txt
* **Шаги выполнения:**

1. Запустить приложение

* **Ожидаемый результат:** Записи загружены в список записей
* **Фактический результат:** Записи загружены в список записей
* **Статус:** Passed

1. **Сохранение данных в формате UTF-8**

* **Идентификатор:** TC-012
* **Название:** Сохранение данных в формате UTF-8
* **Предусловия:** Добавлен заказ с кириллическими символами: "Иван|Тест✅|0|[текущая дата]"
* **Шаги выполнения:**

1. Проверить кодировку файла orders.txt (в приложении блокнот)

* **Ожидаемый результат:** Файл сохранен в UTF-8, символы отображаются корректно
* **Фактический результат:** Файл сохранен в UTF-8, символы отображаются корректно
* **Статус:** Passed

1. **Уведомление при изменении статуса заказа на “Завершен”**

* **Идентификатор:** TC-012
* **Название:** Уведомление при изменении статуса заказа на "Завершён"
* **Предусловия:** В системе есть заказ со статусом "Новый" или "В обработке"
* **Шаги выполнения:**

1. Выбрать первый заказ в списке заказов, где статус заказа “Новый” или “В обработке”
2. Оставить галочку на флажке “Завершен” в поле “Включить уведомления для статусов:”
3. В выпадающем списке “Статус” выбрать “Завершен”
4. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Появляется уведомление с текстом: "Заказ '[имя клиента] - [описание заказа] [время создания]' завершён"
* **Фактический результат:** Появляется уведомление с текстом: "Заказ '[имя клиента] - [описание заказа] [время создания]' завершён"
* **Статус:** Passed

1. **Уведомление при изменении статуса заказа на "В обработке"**

* **Идентификатор:** TC-013
* **Название:** Уведомление при изменении статуса заказа на "В обработке"
* **Предусловия:** В системе есть заказ со статусом "Новый"
* **Шаги выполнения:**

1. Выбрать первый заказ в списке заказов со статусом в “В обработке”
2. Оставить галочку на флажке “В обработке” в поле “Включить уведомления для статусов:”
3. В выпадающем списке “Статус” выбрать “В обработке”
4. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Появляется уведомление с текстом: "Заказ '[имя клиента] - [описание заказа] [время создания]' взят в обработку"
* **Фактический результат:** Появляется уведомление с текстом: "Заказ '[имя клиента] - [описание заказа] [время создания]' взят в обработку"
* **Статус:** Passed

1. **Отсутствие уведомления при отключенной настройке статуса**

* **Идентификатор:** TC-014
* **Название:** Отсутствие уведомления при отключенной настройке статуса
* **Предусловия:** В системе есть заказ со статусом "Новый"
* **Шаги выполнения:**

1. Убрать галочку на флажке “В обработке” в поле “Включить уведомления для статусов”
2. Выбрать первый заказ в списке заказов со статусом “Новый”
3. В выпадающем списке “Статус” выбрать “В обработке”
4. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Уведомление не появляется
* **Фактический результат:** Уведомление не появляется
* **Статус:** Passed

1. **Уведомления не появляются при том же статусе**

* **Идентификатор:** TC-015
* **Название:** Уведомления не появляются при том же статусе
* **Предусловия:** В системе есть заказ со статусом "В обработке"
* **Шаги выполнения:**

1. Оставить галочку на флажке “В обработке” в поле “Включить уведомления для статусов:”
2. Выбрать первый заказ в списке заказов со статусом “В обработке”
3. В выпадающем списке “Статус” выбрать “В обработке”
4. Нажать на кнопку “Обновить статус”

* **Ожидаемый результат:** Уведомление не появляется
* **Фактический результат:** Уведомление не появляется
* **Статус:** Passed