using Microsoft.VisualStudio.TestTools.UnitTesting;

using Moq;

using MegaSklad;

using MegaSklad.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Supabase.Interfaces;

using Supabase.Realtime;

using Supabase.Storage;

using System.Windows;

using System.Windows.Controls;

using System.ComponentModel;

using System.Globalization;

namespace UnitTestProject1

{

[TestClass]

public class CoreLogicTests

{

// TovarIdToNameConverter: Возвращает "Неизвестный товар", если ID отсутствует в кэше

[TestMethod]

public void TovarIdToNameConverter\_UnknownId\_ReturnsDefault()

{

var converter = new TovarIdToNameConverter();

Guid unknownId = Guid.NewGuid();

object result = converter.Convert(unknownId, typeof(string), null, CultureInfo.InvariantCulture);

Assert.AreEqual("Неизвестный товар", result);

}

// MultiImageConverterProfile: Возвращает дефолтное изображение, если id фото отсутствует

[TestMethod]

public void MultiImageConverterProfile\_NullPhotoId\_ReturnsDefaultImage()

{

var converter = new MultiImageConverterProfile();

object result = converter.Convert(new object[] { null, "photoprofil", "user.png" }, typeof(object), null, CultureInfo.InvariantCulture);

Assert.IsInstanceOfType(result, typeof(System.Windows.Media.Imaging.BitmapImage));

}

// MultiImageConverterProfile: Передача некорректных параметров

[TestMethod]

[ExpectedException(typeof(NullReferenceException))]

public void MultiImageConverterProfile\_IncorrectParameters()

{

var converter = new MultiImageConverterProfile();

converter.Convert(new object[] { Guid.NewGuid(), "text1", "text2" }, typeof(object), null, null);

}

// OtchetPoOstatkamItem: Проверка на создание нового отчета, если параметры соответствуют

[TestMethod]

public void OtchetPoOstatkamItem\_Creation()

{

string name = "TestName";

int kol = 10;

var otchet = new OtchetPoOstatkamItem

{

название\_товара = name,

количество = kol,

};

Assert.AreEqual(name, otchet.название\_товара);

Assert.AreEqual(kol, otchet.количество);

}

// ТоварыЗаказаПоставщикуViewModel: Проверка на заполнение параметров

[TestMethod]

public void ТоварыЗаказаПоставщикуViewModel\_CheckParams()

{

Guid idTovara = Guid.NewGuid();

string nazvaineTovara = "Название товара";

int kol = 12;

var tovary = new ТоварыЗаказаПоставщикуViewModel()

{

id\_товара = idTovara,

название\_товара = nazvaineTovara,

количество = kol,

};

Assert.AreEqual(nazvaineTovara, tovary.название\_товара);

Assert.AreEqual(kol, tovary.количество);

Assert.AreEqual(idTovara, tovary.id\_товара);

}

// EditClientWindow IsValidEmail: проверка валидного e-mail

[TestMethod]

public void EditClientWindow\_IsValidEmail\_CorrectName()

{

EditClientWindow editClientWindow = new EditClientWindow(new Клиенты());

bool correct = (bool)editClientWindow.GetType().GetMethod("IsValidEmail", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance).Invoke(editClientWindow, new object[] { "admin@gmail.com" });

Assert.IsTrue(correct);

}

// EditClientWindow IsValidEmail: проверка не валидного e-mail

[TestMethod]

public void EditClientWindow\_InValidEmail\_IncorrectName()

{

EditClientWindow editClientWindow = new EditClientWindow(new Клиенты());

bool incorrect = (bool)editClientWindow.GetType().GetMethod("IsValidEmail", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance).Invoke(editClientWindow, new object[] { "admingmail.com" });

Assert.IsFalse(incorrect);

}

// EditPostavshikWindow IsValidPhone: проверка валидного номера телефона

[TestMethod]

public void EditPostavshikWindow\_IsValidNumber\_CorrectNumber()

{

EditPostavshikWindow editPostavshikWindow = new EditPostavshikWindow(new Поставщики());

bool correct = (bool)editPostavshikWindow.GetType().GetMethod("IsValidPhoneNumber", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance).Invoke(editPostavshikWindow, new object[] { "89513835480" });

Assert.IsTrue(correct);

}

// EditPostavshikWindow IsValidPhone: проверка валидного номера телефона

[TestMethod]

public void EditPostavshikWindow\_IsIValidNumber\_IncorrectNumber()

{

EditPostavshikWindow editPostavshikWindow = new EditPostavshikWindow(new Поставщики());

bool correct = (bool)editPostavshikWindow.GetType().GetMethod("IsValidPhoneNumber", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance).Invoke(editPostavshikWindow, new object[] { "9513835480" });

Assert.IsFalse(correct);

}

// Проверка correct Model

[TestMethod]

public void checkCorrectModel\_isCorrect()

{

ПроверкаДоставки prov = new ПроверкаДоставки();

string s = "Test";

string t = "123";

prov.TextAction = s;

prov.TextComment = t;

Assert.AreEqual("Test", prov.TextAction);

Assert.AreEqual("123", prov.TextComment);

}

}

internal class ПроверкаДоставки

{

public string TextAction { get; set; }

public string TextComment { get; set; }

}

public class OtchetPoOstatkamItem

{

public string название\_товара { get; set; }

public int количество { get; set; }

}

public class ТоварыЗаказаПоставщикуViewModel

{

public Guid id\_товара { get; set; }

public string название\_товара { get; set; }

public int количество { get; set; }

}

}

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 1 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | TovarIdToNameConverter: Возвращает "Неизвестный товар", если ID отсутствует в кэше |
| Резюме испытания | Проверка возвращаемого значения при отсутствии ID товара. |
| Шаги тестирования | 1. Создать экземпляр TovarIdToNameConverter. 2. Вызвать метод Convert с произвольным Guid. |
| Данные тестирования | Случайный Guid |
| Ожидаемый результат | Метод Convert возвращает строку "Неизвестный товар". |
| Фактический результат | Строка "Неизвестный товар" |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 2 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | MultiImageConverterProfile: Возвращает дефолтное изображение, если id фото отсутствует |
| Резюме испытания | Проверка загрузки изображения по умолчанию, если PhotoId = null. |
| Шаги тестирования | 1. Создать экземпляр MultiImageConverterProfile. 2. Вызвать метод Convert с null PhotoId. |
| Данные тестирования | null в качестве PhotoId, пути к ресурсам |
| Ожидаемый результат | Метод Convert возвращает объект BitmapImage. |
| Фактический результат | Объект BitmapImage |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 3 |
| Приоритет теста | Средний |
| Название тестирования/Имя | MultiImageConverterProfile: Передача некорректных параметров |
| Резюме испытания | Тест проверяет брос исключения при некорректных параметрах. |
| Шаги тестирования | 1. Создать экземпляр MultiImageConverterProfile. 2. Передать некорректные параметры в метод Convert |
| Данные тестирования | Передача Object [] object в метод Convert. |
| Ожидаемый результат | Возвращается исключение NullReferenceException. |
| Фактический результат | Сработало исключение NullReferenceException. |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 4 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | OtchetPoOstatkamItem: Проверка на создание нового отчета, если параметры соответствуют |
| Резюме испытания | Проверка успешного создания экземпляра класса OtchetPoOstatkamItem. |
| Шаги тестирования | 1. Создать экземпляр класса OtchetPoOstatkamItem. 2. Установить значения свойств "название\_товара" и "количество". |
| Данные тестирования | "TestName" - строка, 10 - число |
| Ожидаемый результат | Свойства экземпляра класса содержат установленные значения. |
| Фактический результат | Соответствует ожидаемым значениям. |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 5 |
| Приоритет теста | Средний |
| Название тестирования/Имя | ТоварыЗаказаПоставщикуViewModel: Проверка на заполнение параметров |
| Резюме испытания | Проверка свойств класса ТоварыЗаказаПоставщикуViewModel. |
| Шаги тестирования | 1. Создать экземпляр класса ТоварыЗаказаПоставщикуViewModel. 2. Установить значения для id\_товара, название\_товара, количество. |
| Данные тестирования | Guid, "Название товара", int. |
| Ожидаемый результат | Свойства экземпляра класса содержат установленные значения. |
| Фактический результат | Соответствует ожидаемым значениям. |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 6 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | EditClientWindow IsValidEmail: проверка валидного e-mail |
| Резюме испытания | Проверка валидного email. |
| Шаги тестирования | 1. Создать экземпляр класса EditClientWindow. 2. Вызвать метод "IsValidEmail" с валидным email. |
| Данные тестирования | "[admin@gmail.com](https://www.google.com/url?sa=E&q=mailto%3Aadmin%40gmail.com)" |
| Ожидаемый результат | Метод возвращает True. |
| Фактический результат | True |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 7 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | EditClientWindow IsValidEmail: проверка non валидного e-mail |
| Резюме испытания | Проверка невалидного email. |
| Шаги тестирования | 1. Создать экземпляр класса EditClientWindow. 2. Вызвать метод "IsValidEmail" с невалидным email. |
| Данные тестирования | "admingmail.com" |
| Ожидаемый результат | Метод возвращает False. |
| Фактический результат | False |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 8 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | EditPostavshikWindow IsValidPhone: проверка валидного номера телефона |
| Резюме испытания | Проверка валидного номера телефона. |
| Шаги тестирования | 1. Создать экземпляр класса EditPostavshikWindow. 2. Вызвать метод "IsValidPhoneNumber" с валидным номером телефона. |
| Данные тестирования | "89513835480" |
| Ожидаемый результат | Метод возвращает True. |
| Фактический результат | True |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 9 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | EditPostavshikWindow IsIValidPhone: проверка невалидного номера телефона |
| Резюме испытания | Проверка невалидного номера телефона. |
| Шаги тестирования | 1. Создать экземпляр класса EditPostavshikWindow. 2. Вызвать метод "IsValidPhoneNumber" с невалидным номером телефона. |
| Данные тестирования | "9513835480" |
| Ожидаемый результат | Метод возвращает False. |
| Фактический результат | False |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Fail |
| Комментарии | Ожидаемый провал. |

|  |  |
| --- | --- |
| Раздел | Описание |
| TestCase # | 10 |
| Приоритет теста | Средний |
| Название тестирования/Имя | checkCorrectModel\_isCorrect Проверка correct Model |
| Резюме испытания | Проверка CorrectModel |
| Шаги тестирования | 1. Вызвать метод, который возвращает текст Coment и Action |
| Данные тестирования | Выполнение кода возвращает строки Comment и Action |
| Ожидаемый результат | Метод возвращает имя |
| Фактический результат | Ошибка |
| Предпосылки | Скомпилирован и запущен тестовый метод. |
| Постусловия | Нет |
| Статус (Pass/Fail) | Fail |
| Комментарии | Ожидаемый провал. |

