Тестирование программного продукта

**Задание**

Необходимо разработать позитивные и негативные тест-кейсы (не менее 3) для тестирования программных модулей. К ним необходимо разработать юнит-тесты.

Отчёт должен содержать:

1. Таблицы с тест-кейсами с пояснениями, что тестируем;
2. Юнит-тесты с пояснениями (код и результат проверки).

Таблица 1 - Таблица с тест-кейсами с пояснениями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** | **Пройден/не пройден** |
| Способность программы осуществлять расчет скидки для партнера | * запустить программу * перейти на страницу «Партнеры» * в таблице «Партнеры» нажать на первую строку | Появляется label предлагающий скидку в 5%. | Появляется label предлагающий скидку в 5%. Соответствует ожидаемым результатам. | Пройден |
| Способность программы записывать данные о поставщиках в базу данных | * запустить программу * перейти на страницу «Поставщики» * ввести в нужные поля данные «6, 123, Москва, 123» * Нажать кнопку «Сохранить» | В таблице и базе данных появляется поставщик с введенными полями. | В таблице и базе данных появляется поставщик с введенными полями. Соответствует ожидаемым результатам. | Пройден |
| Способность программы удалять записи о сделках | * запустить программу * перейти на страницу «Сделки» * Выделить строчку в таблице «Сделки» * Нажать на кнопку удалить | Запись удалена в таблице и в базе данных. | Запись удалена в таблице и в базе данных.  Соответствует ожидаемым результатам. | Пройден |
| Способность программы уведомлять об ошибке при вводе повторяющегося первичного ключа | * запустить программу * перейти на страницу «Поставщики» * ввести в нужные поля данные «1, 123, Москва, 123» * Нажать кнопку «Сохранить» | Сообщение об ошибке «На столбец ID\_поставщика наложено ограничение по уникальности. Значение «1» уже имеется.». | Сообщение об ошибке «На столбец ID\_поставщика наложено ограничение по уникальности. Значение «1» уже имеется.».  Соответствует ожидаемым результатам. | Пройден |
| Способность программы уведомлять об ошибке при незаполненных полях | * запустить программу * перейти на страницу «Поставщики» * ввести в первое поле данные «7», остальные поля проигнорировать * Нажать кнопку «Сохранить» | Сообщение: «В столбце «Наименование» не допускаются значения NULL.» | Сообщение: «В столбце «Наименование» не допускаются значения NULL.» | Пройден |
| Способность программы уведомлять об ошибке при попытке ввести несуществующее значение внешнего ключа | * запустить программу * перейти на страницу «Партнеры» * ввести в поле «Тип партнеры» данные «буквы» | Система не дает перейти к заполнению следующего поля | Система не дает перейти к заполнению следующего поля. Соответствует ожидаемым результатам. | Пройден |

Юнит-тесты с пояснениями (код и результат проверки)

Код тестов:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace LastDAmnTime.Tests

{

[TestClass]

public class PartnerDiscountTests

{

private string connectionString = "Data Source=ADCLG1;Initial Catalog=ProninaLastLast;Integrated Security=True;Encrypt=False";

// ТЕСТ 1: Расчет скидки 5% для партнера

[TestMethod]

public void CalculateDiscount\_ForFirstPartner\_Returns5Percent()

{

var form = new партнеры1();

form.партнерыDataGridView.Rows[0].Selected = true;

StringAssert.Contains(form.LabelDiscount.Text, "5%");

}

}

[TestClass]

public class SupplierTests

{

private string connectionString = "Data Source=ADCLG1;Initial Catalog=ProninaLastLast;Integrated Security=True;Encrypt=False";

// ТЕСТ 2: Добавление поставщика

[TestMethod]

public void AddSupplier\_ValidData\_SavesToDatabase()

{

var form = new поставщик1();

int testId = 999;

AddTestSupplier(testId, "1234567890", "Москва", "123");

bool exists = CheckSupplierExists(testId);

Assert.IsTrue(exists, "Поставщик должен быть добавлен в базу данных");

DeleteTestSupplier(testId);

}

// ТЕСТ 3: Ошибка при дублировании первичного ключа

[TestMethod]

public void AddSupplier\_DuplicateId\_ThrowsException()

{

int existingId = 1;

var ex = Assert.ThrowsException<SqlException>(() =>

AddTestSupplier(existingId, "123", "Москва", "123"));

StringAssert.Contains(ex.Message, "ограничение по уникальности");

}

// ТЕСТ 4: Ошибка при незаполненных полях

[TestMethod]

public void AddSupplier\_MissingRequiredFields\_ThrowsException()

{

int testId = 998;

var ex = Assert.ThrowsException<SqlException>(() =>

AddTestSupplier(testId, null, null, null));

StringAssert.Contains(ex.Message, "NULL");

}

private void AddTestSupplier(int id, string inn, string address, string contacts)

{

string query = "INSERT INTO Поставщики (ID\_поставщика, ИНН, Юридический\_адрес, Контактные\_данные) " +

"VALUES (@Id, @Inn, @Address, @Contacts)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", id);

command.Parameters.AddWithValue("@Inn", (object)inn ?? DBNull.Value);

command.Parameters.AddWithValue("@Address", (object)address ?? DBNull.Value);

command.Parameters.AddWithValue("@Contacts", (object)contacts ?? DBNull.Value);

command.ExecuteNonQuery();

}

}

}

private bool CheckSupplierExists(int id)

{

string query = "SELECT COUNT(\*) FROM Поставщики WHERE ID\_поставщика = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", id);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

private void DeleteTestSupplier(int id)

{

string query = "DELETE FROM Поставщики WHERE ID\_поставщика = @Id";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", id);

command.ExecuteNonQuery();

}

}

}

}

[TestClass]

public class DealTests

{

private string connectionString = "Data Source=ADCLG1;Initial Catalog=ProninaLastLast;Integrated Security=True;Encrypt=False";

// ТЕСТ 5: Удаление сделки

[TestMethod]

public void DeleteDeal\_ValidDeal\_RemovesFromDatabase()

{

int testDealId = 999;

AddTestDeal(testDealId);

DeleteTestDeal(testDealId);

bool exists = CheckDealExists(testDealId);

Assert.IsFalse(exists, "Сделка должна быть удалена из базы данных");

}

private void AddTestDeal(int dealId)

{

string query = "INSERT INTO История\_партнера (ID\_сделки, ID\_партнера, ID\_сотрудника, Сумма\_сделки, Дата\_сделки) " +

"VALUES (@DealId, 1, 1, 10000, GETDATE())";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@DealId", dealId);

command.ExecuteNonQuery();

}

}

}

private void DeleteTestDeal(int dealId)

{

string query = "DELETE FROM История\_партнера WHERE ID\_сделки = @DealId";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@DealId", dealId);

command.ExecuteNonQuery();

}

}

}

private bool CheckDealExists(int dealId)

{

string query = "SELECT COUNT(\*) FROM История\_партнера WHERE ID\_сделки = @DealId";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@DealId", dealId);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

}

[TestClass]

public class ForeignKeyTests

{

// ТЕСТ 6: Проверка внешнего ключа

[TestMethod]

public void AddPartner\_InvalidType\_ThrowsException()

{

int invalidTypeId = 9999;

var ex = Assert.ThrowsException<SqlException>(() =>

AddTestPartnerWithInvalidType(invalidTypeId));

StringAssert.Contains(ex.Message, "foreign key");

}

private void AddTestPartnerWithInvalidType(int typeId)

{

string query = "INSERT INTO Партнеры (ID\_партнера, Тип, Наименование, Юридический\_адрес, ИНН, ФИО\_директора, Телефон, Почта) " +

"VALUES (9999, @TypeId, 'Test', 'Address', '123', 'Director', '123', 'test@test.com')";

using (SqlConnection connection = new SqlConnection("Data Source=ADCLG1;Initial Catalog=ProninaLastLast;Integrated Security=True;Encrypt=False"))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@TypeId", typeId);

command.ExecuteNonQuery();

}

}

}

}

}

Результаты выполнения тестов:

