**Unit – test к модулю Меню**

1. GetBurgersAsync\_ReturnsMenuItems\_WhenSuccess

Цель: Проверить, что метод возвращает список пицц при успешном запросе.

Мок: GetMenuItems() возвращает тестовый список из 2 элементов.

Проверка: Результат содержит 2 элемента.

2. GetBurgersAsync\_ReturnsEmptyList\_WhenException

Цель: Убедиться, что при ошибке возвращается пустой список.

Мок: GetMenuItems() выбрасывает исключение.

Проверка: Результат пуст (Count = 0).

3. GetImageUrl\_ReturnsOriginalUrl\_ForHttpPath

Цель: Проверить, что HTTP-ссылки возвращаются без изменений.

Мок: GetImageUrl() возвращает исходный URL.

Проверка: Возвращенный URL совпадает с исходным.

4. GetImageUrl\_GeneratesSignedUrl\_ForValidPath

Цель: Убедиться, что для локальных путей генерируется подписанный URL.

Мок: GetImageUrl() возвращает подписанный URL.

Проверка: Результат совпадает с ожидаемым подписанным URL.

Структура:

[TestClass]

public class MenuServiceTests

{

private readonly Mock<ISupabaseMenuWrapper> \_wrapperMock;

private readonly MenuService \_menuService;

public MenuServiceTests()

{

\_wrapperMock = new Mock<ISupabaseMenuWrapper>();

\_menuService = new MenuService(\_wrapperMock.Object);

}

[TestMethod]

public async Task GetBurgersAsync\_ReturnsMenuItems\_WhenSuccess()

{

// Arrange

var expectedItems = new List<MenuItem>

{

new MenuItem { Id = "1", Name = "Pizza 1", Price = 10.99m },

new MenuItem { Id = "2", Name = "Pizza 2", Price = 12.99m }

};

\_wrapperMock.Setup(x => x.GetMenuItems())

.ReturnsAsync(expectedItems);

// Act

var result = await \_menuService.GetBurgersAsync();

// Assert

Assert.AreEqual(2, result.Count);

}

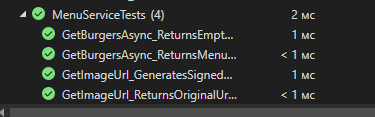
****

Рисунок 1 – Пройденные тесты

**Unit – test к модулю Корзина**

Тесты проверяют корректность работы методов CartService, включая:

1. Добавление/удаление товаров
2. Обновление количества
3. Оформление заказа
4. Обработку изображений
5. Уведомления об изменениях

1. Тесты управления содержимым корзины

AddItem\_ShouldAddNewItem\_WhenItemNotInCart

Проверяет добавление нового товара в пустую корзину.

Ожидание: Товар появляется в корзине с количеством 1.

AddItem\_ShouldIncreaseQuantity\_WhenItemAlreadyInCart

Проверяет увеличение количества при повторном добавлении того же товара.

Ожидание: Количество товара увеличивается до 2.

RemoveItem\_ShouldRemoveItem\_WhenQuantityIsOne

Проверяет удаление товара, если его количество равно 1.

Ожидание: Корзина становится пустой.

RemoveItem\_ShouldDecreaseQuantity\_WhenQuantityMoreThanOne

Проверяет уменьшение количества товара (но не удаление, если количество > 1).

Ожидание: Количество уменьшается с 2 до 1.

AddOrUpdateItem\_ShouldAddItem\_WithSpecificQuantity

Проверяет добавление товара с указанным количеством (например, 3).

Ожидание: Товар добавляется с заданным количеством.

AddOrUpdateItem\_ShouldRemoveItem\_WhenQuantityBecomesZero

Проверяет удаление товара, если количество становится 0 или меньше.

Ожидание: Товар исчезает из корзины.

Clear\_ShouldRemoveAllItems

Проверяет полную очистку корзины.

Ожидание: Корзина пуста после вызова Clear().

2. Тесты обработки изображений

AddItemWithImage\_ShouldUsePlaceholder\_WhenImageUrlIsEmpty

Проверяет подстановку заглушки (placeholder), если у товара нет изображения.

Ожидание: В корзину добавляется товар с URL /images/pizza-placeholder.png.

3. Тесты оформления заказа (Checkout)

Checkout\_ShouldReturnFailure\_WhenOrderCreationFails

Проверяет обработку ошибки при создании заказа (например, сбой БД).

Ожидание: Возвращается success = false и orderId = Guid.Empty.

4. Тест уведомлений об изменениях

NotifyStateChanged\_ShouldInvokeOnChange

Проверяет, что событие OnChange вызывается при изменении корзины (например, добавлении товара).

Ожидание: Флаг eventFired становится true.

[TestClass]

public class CartServiceTests

{

private readonly Mock<ISupabaseCartWrapper> \_wrapperMock;

private readonly CartService \_cartService;

public CartServiceTests()

{

\_wrapperMock = new Mock<ISupabaseCartWrapper>();

\_cartService = new CartService(\_wrapperMock.Object);

}

[TestMethod]

public void AddItem\_ShouldAddNewItem\_WhenItemNotInCart()

{

// Arrange

var item = new MenuItem { Id = "1", Name = "Margherita" };

// Act

\_cartService.AddItem(item);

// Assert

Assert.AreEqual(1, \_cartService.Items.Count);

Assert.AreEqual("Margherita", \_cartService.Items[0].Item.Name);

Assert.AreEqual(1, \_cartService.Items[0].Quantity);

}

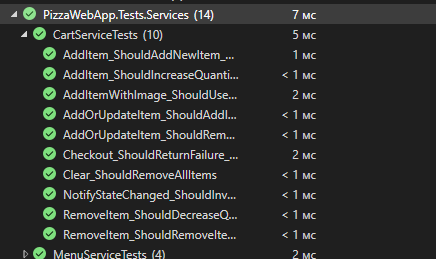


Рисунок 2 – Пройденные тесты

**Unit – test к модулю Информация о заказе**

Тесты проверяют работу страницы, отображающей детали заказа после его оформления, включая:

Загрузку информации о заказе

Форматирование времени

Навигацию

Вспомогательные функции

1. Тесты загрузки данных заказа

OnInitializedAsync\_ShouldLoadOrderDetails\_WhenOrderExists

Проверяет загрузку деталей заказа (например, списка пицц) при корректном orderId.

Ожидание:

OrderDetails содержит 1 элемент с названием "Pepperoni" и количеством 2.

Сервис IOrderService вызывается с переданным orderId.

2. Тесты форматирования времени

FormatMoscowTime\_ShouldAdd3Hours

Проверяет конвертацию UTC-времени в московское (UTC+3).

Ожидание:

Время 10:00 преобразуется в 13:00.

3. Тесты навигации

GoToMenu\_ShouldNavigateToMenuPage

Проверяет переход на страницу меню при вызове метода GoToMenu().

Ожидание:

NavigationManager перенаправляет на /menu.

[TestClass]

public class OrderConfirmedTests

{

private Mock<IOrderService> \_orderServiceMock;

private TestNavigationManager \_navigationManager;

private OrderConfirmed \_component;

public class OrderConfirmed

{

public IOrderService OrderService { get; set; }

public NavigationManager Navigation { get; set; }

public List<OrderDetail> OrderDetails { get; set; } = new();

public string OrderId { get; set; }

public async Task OnInitializedAsync()

{

if (Guid.TryParse(OrderId, out var orderGuid))

{

var details = await OrderService.GetOrderDetailsAsync(orderGuid);

if (details != null)

{

OrderDetails = details;

}

}

}

public string FormatMoscowTime(DateTime dateTime)

{

return dateTime.AddHours(3).ToString("HH:mm");

}

public void GoToMenu()

{

Navigation.NavigateTo("/menu");

}

public void PrintOrder()

{

Console.WriteLine("Печать заказа...");

}

}

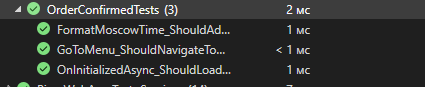


Рисунок 3 – Пройденные тесты