**Государственное бюджетное профессиональное образовательное учреждение**

**«Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Мухина,

\_\_\_\_\_\_\_\_\_\_\_\_ Е.П. Голубева,

«21»\_\_марта\_\_2025 г.

**ОТЧЕТ** **ПО**

**УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Руководитель: \_\_\_\_\_\_\_\_\_\_ Мухина Л.В., Голубева Е.П. 21.03.2025 г.

Студент: \_\_\_\_\_\_\_\_\_\_Климычева Анастасия Артемовна 21.03.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

**Содержание**

[Введение 3](#_Toc192915657)

[Разработка информационной системы «Пиши-Стирай» 4](#_Toc192915658)

[Разработка базы данных, объектов базы данных и импорт 4](#_Toc192915659)

[Разработка настольного приложения 9](#_Toc192915660)

[Разработка библиотеки 14](#_Toc192915661)

[Тестирование библиотеки 15](#_Toc192915662)

[Тестовая документация 17](#_Toc192915663)

[Заключение 20](#_Toc192915664)

[Приложения 21](#_Toc192915665)

[Приложение А 21](#_Toc192915666)

[Приложение Б 24](#_Toc192915667)

[Приложение В 26](#_Toc192915668)

# Введение

Целью учебной практики является разработка трех информационных систем, в соответствии с модулями:

1. Работа с базой данных: разработка, администрирование, подготовка и импорт данных в базу;
2. Разработка модулей программного обеспечения в соответствии с требованиями предметной области и руководством по стилю.
3. Разработка библиотек отдельных методов.
4. Разработка модульных тестов к библиотекам.
5. Написание тестовых сценариев для конкретного функционала, согласно требованиям предметной области, охватывающих различные варианты исходов работы алгоритма.
6. Написание readme для каждого приложения.

# Разработка информационной системы «Пиши-Стирай»

## Разработка базы данных, объектов базы данных и импорт

При помощи имеющегося скрипта созданы основные таблицы, после просмотра данных для импорта база данных была доработана (добавлены сущности, атрибуты и соответствующие связи). Таким образом база данных была приведена в третью нормальную форму. На диаграмме 1 представлена ER-диаграмма, которая отражает сущности и связи разработанной базы данных для ООО «Пиши-Стирай».

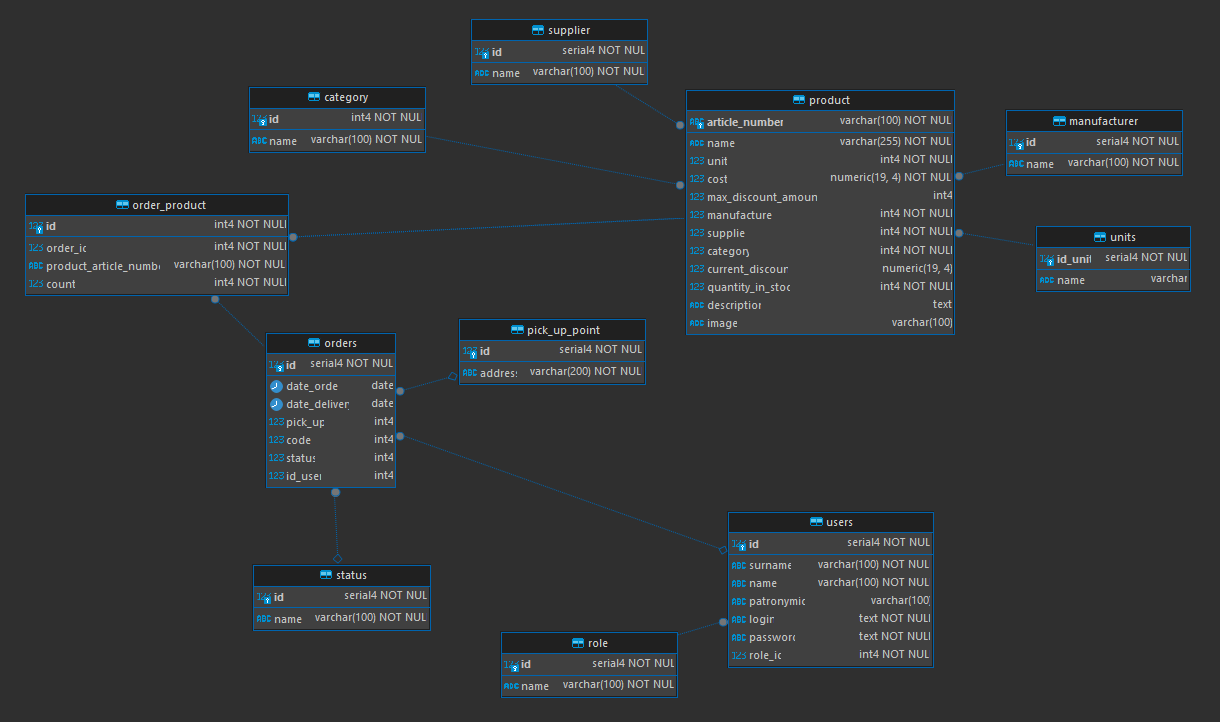


Рисунок №1 – ER-диаграмма базы данных

Скрипт для создания данной базы данных находиться в приложении А Табл. 1. Данные для импорта приведены к необходимому формату в программе Excel. На рисунках 2-12 представлены заполненные таблицы данными для импорта.

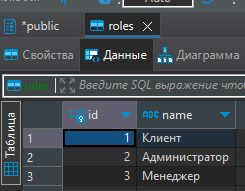


Рисунок №2 – Таблица ролей пользователей

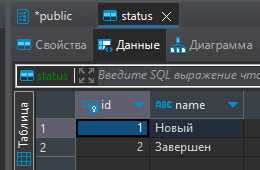


Рисунок №3 – Таблица статусы заказа

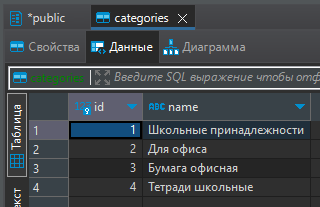


Рисунок №4 – Таблица категорий товаров

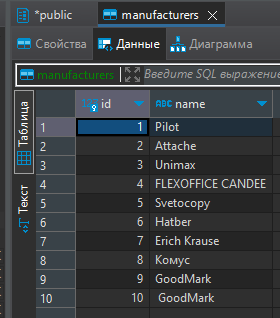


Рисунок №5 – Таблица производителей

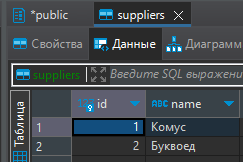


Рисунок №6 – Таблица поставщиков

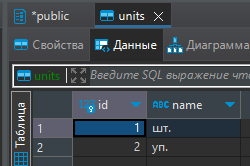


Рисунок №7 – Таблица единиц измерения

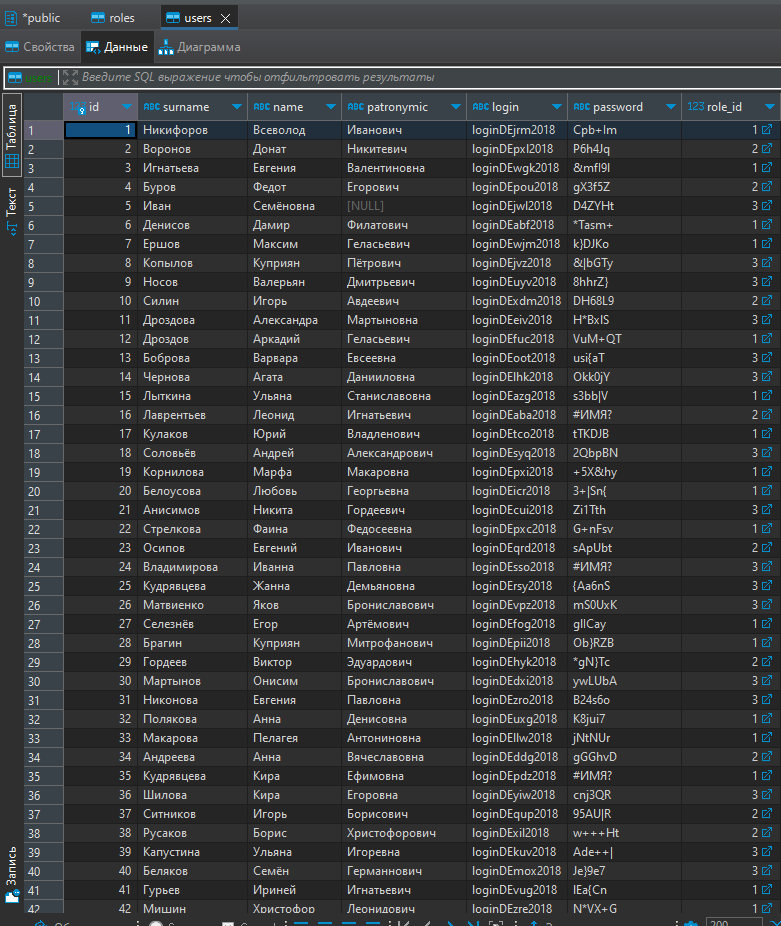


Рисунок №8 – Таблица пользователей

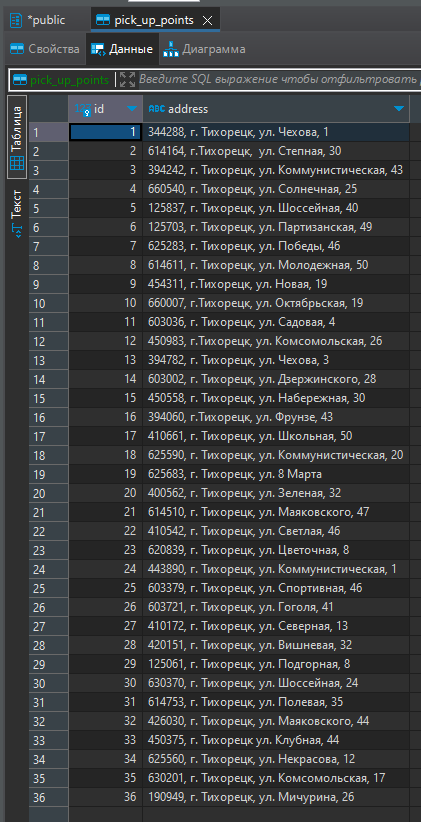


Рисунок №9 – Таблица пунктов выдачи

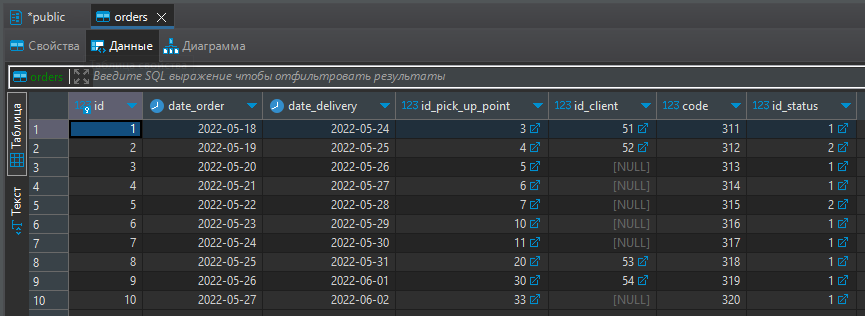


Рисунок №10 – Таблица заказов

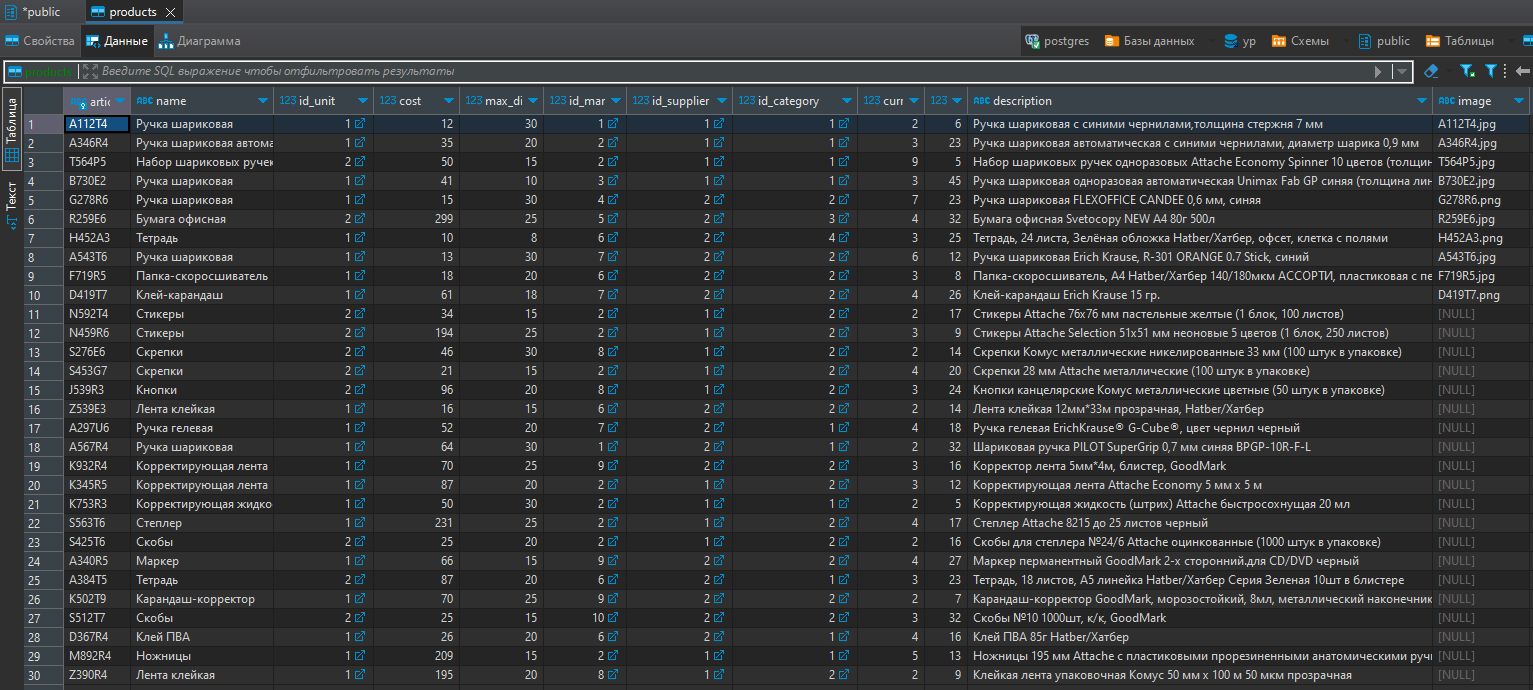


Рисунок №11 – Таблица товаров

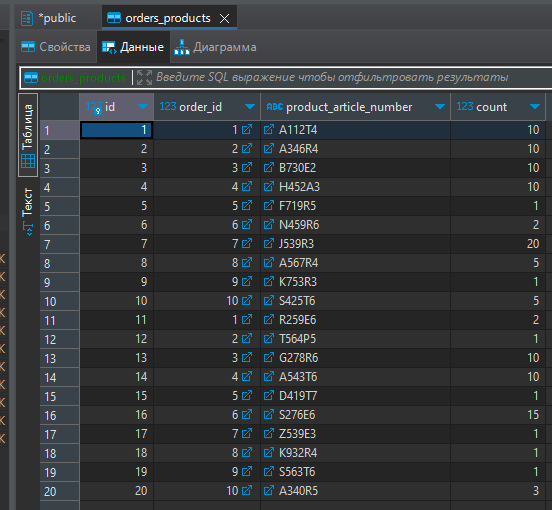


Рисунок №12 – Таблица товаров в заказе

## Разработка настольного приложения

1. Авторизация

В процессе разработки создана страница входа, которая появляется при запуске приложения. На этой странице пользователю предлагается ввести логин и пароль, а также возможность войти в систему в роли гостя.

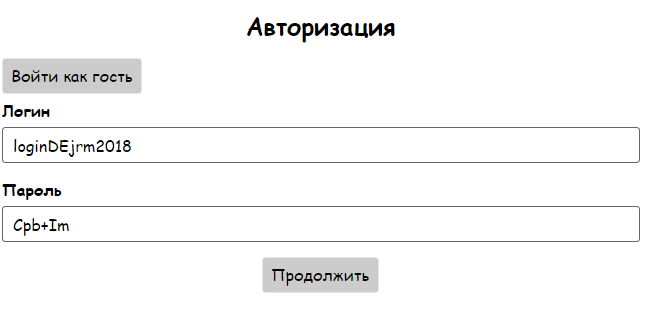


Рисунок №13 – Авторизация

После первой неуспешной попытки авторизации выводится сообщение о неудаче, а также появляется поле для ввода CAPTCHA (состоящей из 4 символов, включающих цифры и буквы латинского алфавита, которые могут быть перечеркнуты).

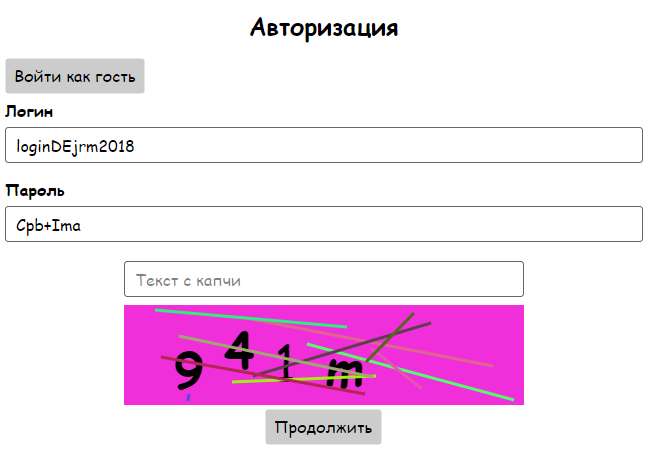


Рисунок №14 – CAPTCHA

После попытки неудачной авторизации с вводом CAPTCHA система блокирует возможность дальнейшего входа на 10 секунд.

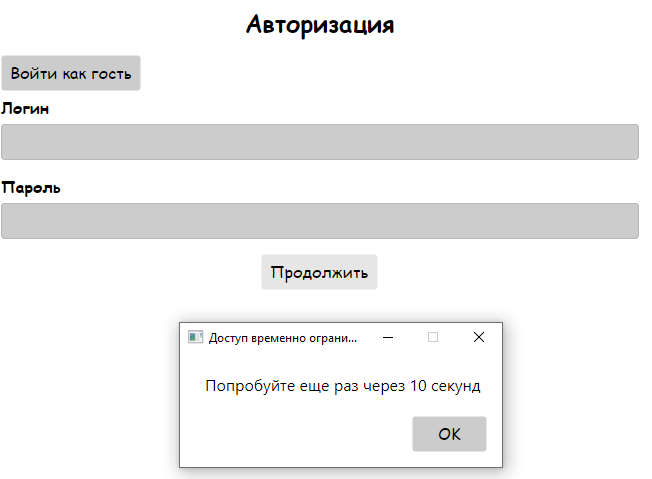


Рисунок №15 – Блокировка

1. Список товаров

Реализован поиск, сортировка и фильтрация списка товаров, происходящая совместно и в реальном времени без нажатия дополнительных кнопок. При нажатии на элемент правой кнопкой мыши появляется контекстное меню с возможностью добавить продукт к заказу в количестве 1 шт. При повторном нажатии количество продукта в заказе увеличится. Если в корзине есть хотя бы один продукт – появляется кнопка «Посмотреть заказ».

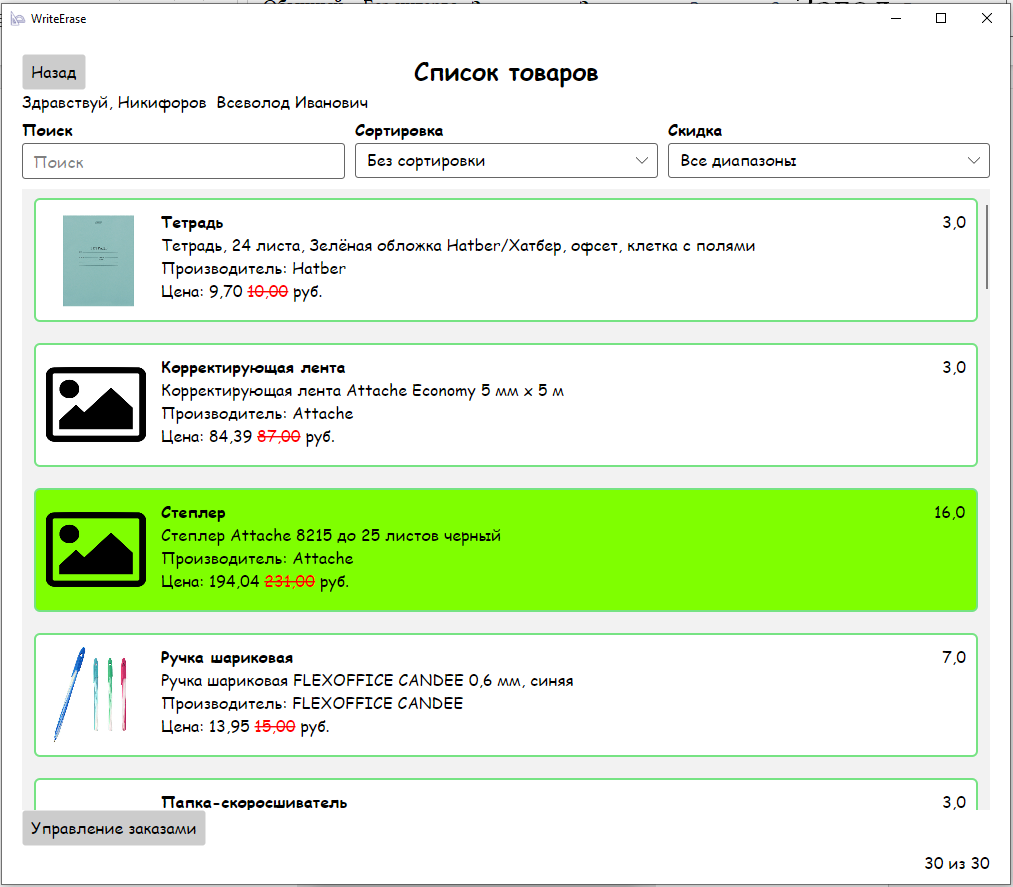


Рисунок №16 – Список продуктов

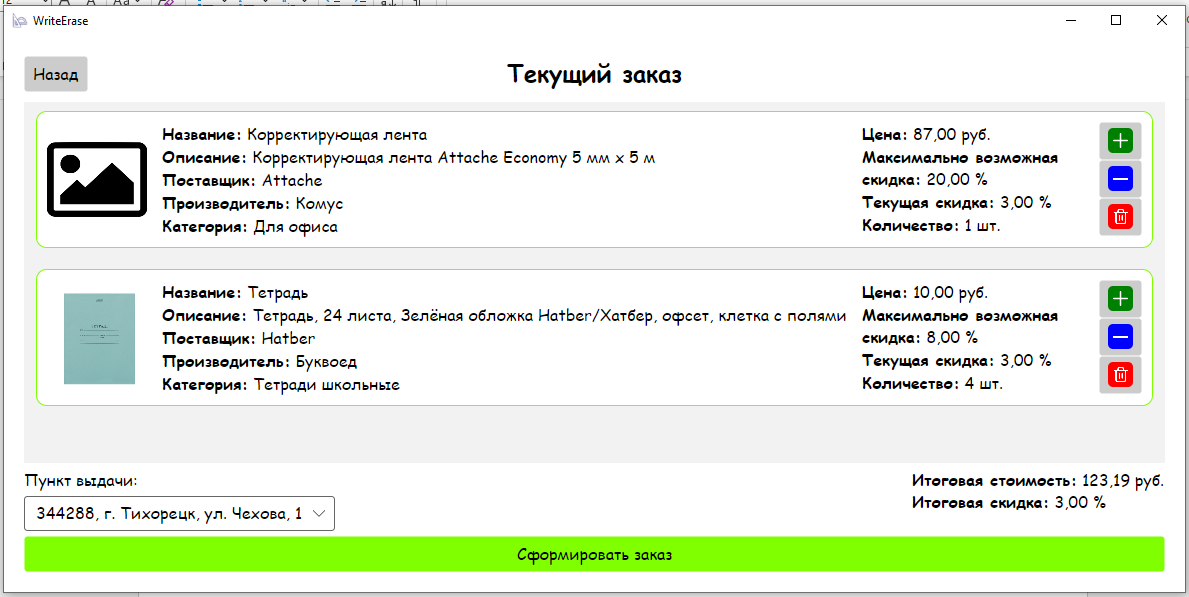


Рисунок №17 – Заказ

Если у товара снижена стоимость, то основная цена перечеркнута, и рядом с ней указана итоговая стоимость. В нижней части окна отображается количество выведенных данных и общее количество записей в базе. В случае, если данные в таблицу выводятся после фильтрации или поиска, количество выведенных данных обновляется исходя из размера выборки.

Для администратора и менеджера в системе реализован функционал работы с заказами. На странице «Управление заказами» присутствует сортировка заказа по убыванию и возрастанию стоимости, а также фильтрация по размеру суммарной скидки. Если все товары в заказе есть на складе в наличии более 3 позиций, то строка с заказом выделена цветом #20b2aa. Если хотя бы одного товара нет на складе, то строка выделена цветом #ff8c00.

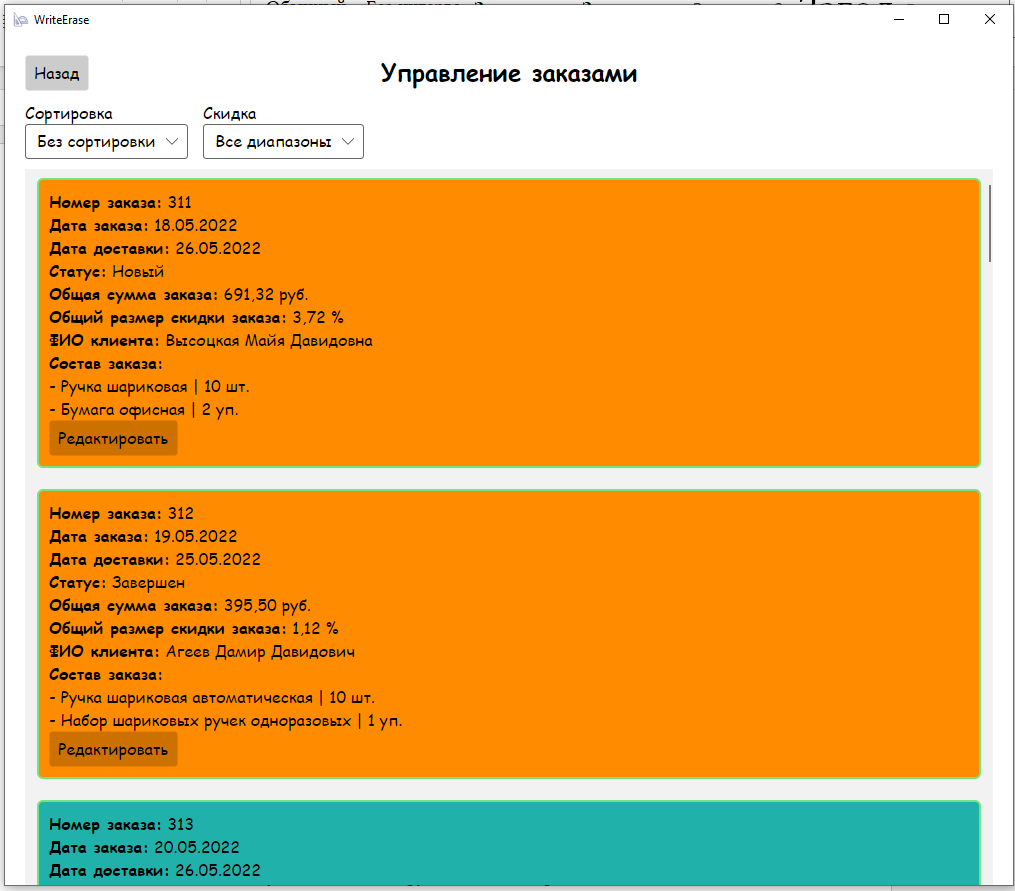


Рисунок №18 – Заказы клиентов

При нажатии на заказ открывается окно изменения заказа, в котором возможно поменять дату доставки и статус заказа (выпадающий список со статусами заказа из базы данных). При изменении информации о заказе отображается сообщение об изменении данных заказа и осуществляется переход на страницу всех заказов.

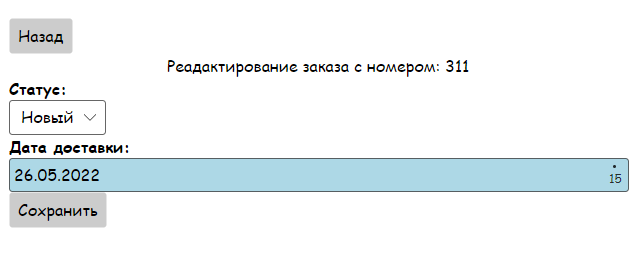


Рисунок №19 – Изменение заказа

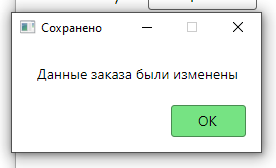
****

Рисунок №20 – Сообщение о сохранении изменений

## Разработка библиотеки

Библиотека имеет один метод «AvailablePeriods», который принимает такие данные, как массивы начальных времен (startTimes) и их продолжительности (durations), начало и конец рабочего дня (beginWorkingTime, endWorkingTime), продолжительность консультации (consultationTime). На выходе возвращается массив строк, представляющий доступные временные интервалы.

Код библиотеки находиться в Прил. Б, Табл. 1.

В начале метода идет проверка на корректность переданных значений. Далее определяется список занятых интервалов. После этого интервалы сортируются по времени начала для создания списка свободных интервалов. После этого проверяются интервалы от начала рабочего дня до первого занятого интервала, таким образом в лист свободных интервалов добавляются новые интервалы. Полученный список преобразуется в массив строк.

## Тестирование библиотеки

Для разработки библиотеки разрабатывались 10 Unit-тестов, которые охватывают все основные исключения и написаны по технике TDD. Эти тесты можно увидеть в Прил. В. Табл. 1. Вот краткое описание каждого теста:

1. «AvailablePeriods\_NoBusyIntervals\_ReturnsFullDay» проверяет, что если нет занятых интервалов, то возвращаются все возможные интервалы в течение рабочего дня.
2. «AvailablePeriods\_OneBusyInterval\_ReturnsCorrectFreeIntervals» проверяет корректность возвращаемых свободных интервалов при наличии одного занятого интервала в середине дня.
3. «AvailablePeriods\_BusyIntervalAtStart\_ReturnsCorrectFreeIntervals» проверяет корректность возвращаемых свободных интервалов при наличии занятого интервала в начале дня.
4. «AvailablePeriods\_BusyIntervalAtEnd\_ReturnsCorrectFreeIntervals» проверяет корректность возвращаемых свободных интервалов при наличии занятого интервала в конце дня.
5. «AvailablePeriods\_OverlappingBusyIntervals\_ReturnsCorrectFreeIntervals» проверяет корректность возвращаемых свободных интервалов при наличии нескольких пересекающихся занятых интервалов.
6. «AvailablePeriods\_ConsultationTimeTooLong\_ReturnsCorrectInterval» проверяет, если минимальное время консультации больше, чем доступное свободное время, возвращаются корректные интервалы.
7. «AvailablePeriods\_BeginWorkingTimeAfterEnd\_ReturnsError» проверяет, что при некорректных входных данных (начало рабочего дня позже его окончания) возвращается ошибка.
8. «AvailablePeriods\_InvalidConsultationTime\_ReturnsError» проверяет, что при некорректных входных данных (минимальное время <= 0) возвращается ошибка.
9. «AvailablePeriods\_ArraysLengthMismatch\_ReturnsError» проверяет, что при несовпадении длин массивов startTimes и durations возвращается ошибка.
10. «AvailablePeriods\_FullDayBusy\_ReturnsEmptyArray» проверяет, если занятые интервалы покрывают весь день, возвращается пустой массив.

## Тестовая документация

Реализована тестовая документация для тестирования функции добавления товара администратором.

Аннотация тестов

|  |  |
| --- | --- |
| **Название проекта** | «WriteErase» |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Климычева Анастасия Артемовна |
| **Дата(ы) теста** | 14.03.2025 |

Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AP\_1: Full Field Product Creation |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка добавления товара администратором с заполнением всех обязательных полей |
| **Краткое изложение теста** | Тест проверяет возможность добавления нового товара в систему при корректном заполнении всех полей формы. |
| **Этапы теста** | 1. Нажать на кнопку «Добавить товар». 2. Внести данные во все поля формы.   Нажать кнопку «Сохранить». |
| **Тестовые данные** | Данные о товаре |
| **Ожидаемый результат** | * Система отображает сообщение об успешном добавлении товара.   Товар сохраняется в базе данных и появляется в общем списке товаров. |
| **Фактический результат** | * Сообщение об успешном добавлении товара отображается.   Товар добавлен в базу данных и отображается в списке. |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизирован в системе |
| **Постусловие** | Возврат на страницу со списком товаров |
| **Примечания/комментарии** | - |

Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AP\_2: Empty Field Product Addition |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка добавления товара администратором без заполнения полей |
| **Краткое изложение теста** | Тест проверяет реакцию системы при попытке добавить товар с пустыми полями формы. |
| **Этапы теста** | 1. Нажать кнопку «Добавить товар».  2. Нажать кнопку «Сохранить». |
| **Тестовые данные** | Отсутствуют |
| **Ожидаемый результат** | Пользователю отображается сообщение об ошибке: «Проверьте правильность заполнения полей и попробуйте еще раз». |
| **Фактический результат** | Пользователю отображается сообщение об ошибке: «Проверьте правильность заполнения полей и попробуйте еще раз». |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизирован в системе |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Тестовый пример #3:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AP\_3: Negative Price Input Test |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка добавления товара с отрицательной ценой |
| **Краткое изложение теста** | Тест проверяет, позволяет ли система ввести отрицательное значение в поле «Стоимость» при добавлении товара. |
| **Этапы теста** | 1. Нажать кнопку «Добавить товар».  2. Ввести в поле «Стоимость» отрицательное значение. |
| **Тестовые данные** | Данные о товаре (название, описание и т.д.), кроме поля «Стоимость», где указано отрицательное значение. |
| **Ожидаемый результат** | Поле «Стоимость» автоматически очищается и устанавливается значение 0, предотвращая ввод отрицательных чисел. |
| **Фактический результат** | Поле «Стоимость» очищается и отображает значение 0, блокируя ввод отрицательных значений. |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизирован в системе |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Тестовый пример #4:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AP\_4: Missing Product Name Test |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка добавления товара без указания названия |
| **Краткое изложение теста** | Тест проверяет, позволяет ли система добавить товар, если поле «Наименование» осталось пустым. |
| **Этапы теста** | 1. Нажать кнопку «Добавить товар».  2. Ввести данные о товаре, оставив поле «Наименование» пустым.  3. Нажать кнопку «Сохранить». |
| **Тестовые данные** | Данные о товаре (кроме поля «Наименование»). |
| **Ожидаемый результат** | Пользователь видит сообщение об ошибке: «Проверьте правильность заполнения полей и попробуйте еще раз». |
| **Фактический результат** | Пользователь видит сообщение об ошибке: «Проверьте правильность заполнения полей и попробуйте еще раз». |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизирован в системе |
| **Постусловие** | - |
| **Примечания/комментарии** | Поля «Артикул» и «Наименование» обязательны для добавления товара. |

Тестовый пример #5:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_AP\_5: Non-Numeric Input Validation |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка ввода нечисловых значений в числовое поле |
| **Краткое изложение теста** | Тест проверяет, позволяет ли система вводить символы и буквы в поле, предназначенное для числовых значений. |
| **Этапы теста** | 1. Нажать кнопку «Добавить товар».  2. Ввести в поле «Текущая скидка» (или другое поле, принимающее численное значение) символы и буквы |
| **Тестовые данные** | Символы и буквы (например, «abc», «!@#»). |
| **Ожидаемый результат** | Поле автоматически очищается, предотвращая ввод нечисловых значений. |
| **Фактический результат** | Поле очищается, блокируя ввод символов и букв. |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор авторизирован в системе |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

# Заключение

В ходе выполнения задания успешно выполнены все поставленные задачи, соответствующие модулям.

Проведен анализ требований к разработке, администрированию и защите баз данных, что позволило сформировать четкое понимание структуры и функциональности системы.

Разработаны модули программного обеспечения, соответствующие требованиям предметной области и руководствам по стилю.

Описаны тестовые сценарии, охватывающие различные варианты исходов работы алгоритма. Реализованы библиотеки отдельных методов, а также разработаны проекты модульных тестов.

Все результаты работы систематизированы и представлены в репозитории с подробным описанием в файле README.md.

Выполнение данного задания позволило закрепить навыки работы с базами данных, разработки программных модулей, модульного тестирования.

# Приложения

## Приложение А

Таблица №1 – Скрипт БД

|  |
| --- |
| -- public.category определение  CREATE TABLE public.category (  id int4 DEFAULT nextval('categoty\_id\_seq'::regclass) NOT NULL,  "name" varchar(100) NOT NULL,  CONSTRAINT categoty\_pkey PRIMARY KEY (id)  );  -- public.manufacturer определение  CREATE TABLE public.manufacturer (  id serial4 NOT NULL,  "name" varchar(100) NOT NULL,  CONSTRAINT manufacturer\_pkey PRIMARY KEY (id)  );  -- public.pick\_up\_point определение  CREATE TABLE public.pick\_up\_point (  id serial4 NOT NULL,  address varchar(200) NOT NULL,  CONSTRAINT pick\_up\_point\_pkey PRIMARY KEY (id)  );  -- public."role" определение  CREATE TABLE public."role" (  id serial4 NOT NULL,  "name" varchar(100) NOT NULL,  CONSTRAINT role\_pkey PRIMARY KEY (id)  );  -- public.status определение  CREATE TABLE public.status (  id serial4 NOT NULL,  "name" varchar(100) NOT NULL,  CONSTRAINT status\_pkey PRIMARY KEY (id)  );  -- public.supplier определение  CREATE TABLE public.supplier (  id serial4 NOT NULL,  "name" varchar(100) NOT NULL,  CONSTRAINT supplier\_pkey PRIMARY KEY (id)  );  -- public.units определение  CREATE TABLE public.units (  "name" varchar NULL,  id\_unit serial4 NOT NULL,  CONSTRAINT units\_pk PRIMARY KEY (id\_unit)  );  -- public.product определение  CREATE TABLE public.product (  article\_number varchar(100) NOT NULL,  "name" varchar(255) NOT NULL,  unit int4 NOT NULL,  "cost" numeric(19, 4) NOT NULL,  max\_discount\_amount int4 NULL,  manufacturer int4 NOT NULL,  supplier int4 NOT NULL,  category int4 NOT NULL,  current\_discount numeric(19, 4) NULL,  quantity\_in\_stock int4 NOT NULL,  description text NULL,  image varchar(100) NULL,  CONSTRAINT product\_pkey PRIMARY KEY (article\_number),  CONSTRAINT product\_category\_fkey FOREIGN KEY (category) REFERENCES public.category(id),  CONSTRAINT product\_manufacturer\_fkey FOREIGN KEY (manufacturer) REFERENCES public.manufacturer(id),  CONSTRAINT product\_supplier\_fkey FOREIGN KEY (supplier) REFERENCES public.supplier(id),  CONSTRAINT product\_units\_fk FOREIGN KEY (unit) REFERENCES public.units(id\_unit) ON DELETE CASCADE ON UPDATE CASCADE  );  -- public.users определение  CREATE TABLE public.users (  id serial4 NOT NULL,  surname varchar(100) NOT NULL,  "name" varchar(100) NOT NULL,  patronymic varchar(100) NULL,  login text NOT NULL,  "password" text NOT NULL,  role\_id int4 NOT NULL,  CONSTRAINT users\_pkey PRIMARY KEY (id),  CONSTRAINT users\_role\_id\_fkey FOREIGN KEY (role\_id) REFERENCES public."role"(id)  );  -- public.orders определение  CREATE TABLE public.orders (  id serial4 NOT NULL,  date\_order date NULL,  date\_delivery date NULL,  pick\_up int4 NULL,  code int4 NULL,  status int4 NULL,  id\_user int4 NULL,  CONSTRAINT orders\_pkey PRIMARY KEY (id),  CONSTRAINT orders\_pick\_up\_fkey FOREIGN KEY (pick\_up) REFERENCES public.pick\_up\_point(id) ON DELETE CASCADE ON UPDATE CASCADE,  CONSTRAINT orders\_status\_fkey FOREIGN KEY (status) REFERENCES public.status(id),  CONSTRAINT orders\_users\_fk FOREIGN KEY (id\_user) REFERENCES public.users(id) ON DELETE CASCADE ON UPDATE CASCADE  );  -- public.order\_product определение  CREATE TABLE public.order\_product (  id int4 DEFAULT nextval('orderproduct\_id\_seq'::regclass) NOT NULL,  order\_id int4 NOT NULL,  product\_article\_number varchar(100) NOT NULL,  count int4 NOT NULL,  CONSTRAINT orderproduct\_pkey PRIMARY KEY (id),  CONSTRAINT order\_product\_product\_fk FOREIGN KEY (product\_article\_number) REFERENCES public.product(article\_number) ON DELETE CASCADE ON UPDATE CASCADE,  CONSTRAINT orderproduct\_order\_id\_fkey FOREIGN KEY (order\_id) REFERENCES public.orders(id) ON DELETE CASCADE ON UPDATE CASCADE,  CONSTRAINT orderproduct\_product\_article\_number\_fkey FOREIGN KEY (product\_article\_number) REFERENCES public.product(article\_number)  ); |

## Приложение Б

Таблица №1 –Код класса Calculations

|  |
| --- |
| namespace SF2022User\_NN\_Lib  {      public class Calculations      {          public string[] AvailablePeriods(TimeSpan[] startTimes, int[] durations, TimeSpan beginWorkingTime, TimeSpan endWorkingTime, int consultationTime)          {              if (beginWorkingTime >= endWorkingTime)                  return new[] { "Начало рабочего дня должно быть раньше его окончания." };              if (consultationTime <= 0)                  return new[] { "Минимальное время должно быть больше нуля." };              if (startTimes == null || durations == null || startTimes.Length != durations.Length)                  return new[] { "Ошибка: массивы startTimes и durations должны быть не null и одинаковой длины." };              var busyIntervals = CreateBusyIntervals(startTimes, durations);              busyIntervals.Sort((a, b) => a.Start.CompareTo(b.Start));              return GenerateFreeIntervals(busyIntervals, beginWorkingTime, endWorkingTime, consultationTime);          }          private List<(TimeSpan Start, TimeSpan End)> CreateBusyIntervals(TimeSpan[] startTimes, int[] durations)          {              var intervals = new List<(TimeSpan Start, TimeSpan End)>();              for (int i = 0; i < startTimes.Length; i++)              {                  intervals.Add((startTimes[i], startTimes[i].Add(TimeSpan.FromMinutes(durations[i]))));              }              return intervals;          }          private string[] GenerateFreeIntervals(List<(TimeSpan Start, TimeSpan End)> busyIntervals, TimeSpan beginWorkingTime, TimeSpan endWorkingTime, int consultationTime)          {              var freeIntervals = new List<string>();              var currentTime = beginWorkingTime;              while (currentTime.Add(TimeSpan.FromMinutes(consultationTime)) <= endWorkingTime)              {                  var intervalEnd = currentTime.Add(TimeSpan.FromMinutes(consultationTime));                  if (!IsIntervalBusy(currentTime, intervalEnd, busyIntervals))                  {                      freeIntervals.Add($"{currentTime:hh\\:mm}-{intervalEnd:hh\\:mm}");                  }                  currentTime = currentTime.Add(TimeSpan.FromMinutes(consultationTime));              }              return freeIntervals.ToArray();          }          private bool IsIntervalBusy(TimeSpan start, TimeSpan end, List<(TimeSpan Start, TimeSpan End)> busyIntervals)          {              return busyIntervals.Any(busy => start < busy.End && end > busy.Start);          }      }  } |

## Приложение В

Таблица №1 – Тесты класса Calculations

|  |
| --- |
| using SF2022User\_NN\_Lib;  namespace TestProject  {  [TestClass]  public class UnitTestCalculationClass  {  private readonly Calculations \_calculations;  public UnitTestCalculationClass()  {  \_calculations = new Calculations();  }  // Тест 1: Нет занятых интервалов, весь день свободен  [TestMethod]  public void AvailablePeriods\_NoBusyIntervals\_ReturnsFullDay()  {  var startTimes = new TimeSpan[] { };  var durations = new int[] { };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "09:00-09:30", "09:30-10:00", "10:00-10:30", "10:30-11:00", "11:00-11:30", "11:30-12:00", "12:00-12:30", "12:30-13:00", "13:00-13:30", "13:30-14:00", "14:00-14:30", "14:30-15:00", "15:00-15:30", "15:30-16:00", "16:00-16:30", "16:30-17:00" }, result);  }  // Тест 2: Один занятый интервал в середине дня  [TestMethod]  public void AvailablePeriods\_OneBusyInterval\_ReturnsCorrectFreeIntervals()  {  var startTimes = new[] { new TimeSpan(10, 0, 0) };  var durations = new[] { 60 };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "09:00-09:30", "09:30-10:00", "11:00-11:30", "11:30-12:00", "12:00-12:30", "12:30-13:00", "13:00-13:30", "13:30-14:00", "14:00-14:30", "14:30-15:00", "15:00-15:30", "15:30-16:00", "16:00-16:30", "16:30-17:00" }, result);  }  // Тест 3: Занятый интервал в начале дня  [TestMethod]  public void AvailablePeriods\_BusyIntervalAtStart\_ReturnsCorrectFreeIntervals()  {  var startTimes = new[] { new TimeSpan(9, 0, 0) };  var durations = new[] { 60 };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "10:00-10:30", "10:30-11:00", "11:00-11:30", "11:30-12:00", "12:00-12:30", "12:30-13:00", "13:00-13:30", "13:30-14:00", "14:00-14:30", "14:30-15:00", "15:00-15:30", "15:30-16:00", "16:00-16:30", "16:30-17:00" }, result);  }  // Тест 4: Занятый интервал в конце дня  [TestMethod]  public void AvailablePeriods\_BusyIntervalAtEnd\_ReturnsCorrectFreeIntervals()  {  var startTimes = new[] { new TimeSpan(16, 0, 0) };  var durations = new[] { 60 };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "09:00-09:30", "09:30-10:00", "10:00-10:30", "10:30-11:00", "11:00-11:30", "11:30-12:00", "12:00-12:30", "12:30-13:00", "13:00-13:30", "13:30-14:00", "14:00-14:30", "14:30-15:00", "15:00-15:30", "15:30-16:00" }, result);  }  // Тест 5: Несколько занятых интервалов с пересечениями  [TestMethod]  public void AvailablePeriods\_OverlappingBusyIntervals\_ReturnsCorrectFreeIntervals()  {  var startTimes = new[] { new TimeSpan(10, 0, 0), new TimeSpan(10, 30, 0) };  var durations = new[] { 60, 30 };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(  new[] {  "09:00-09:30",  "09:30-10:00",  "11:00-11:30",  "11:30-12:00",  "12:00-12:30",  "12:30-13:00",  "13:00-13:30",  "13:30-14:00",  "14:00-14:30",  "14:30-15:00",  "15:00-15:30",  "15:30-16:00",  "16:00-16:30",  "16:30-17:00"  }, result);  }  // Тест 6: Минимальное время больше, чем доступное свободное время  [TestMethod]  public void AvailablePeriods\_ConsultationTimeTooLong\_ReturnsCorrectIntervals()  {  var startTimes = new[] { new TimeSpan(10, 0, 0) };  var durations = new[] { 60 };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 120;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "11:00-13:00", "13:00-15:00", "15:00-17:00" }, result);  }  // Тест 7: Некорректные входные данные (начало рабочего дня позже конца)  [TestMethod]  public void AvailablePeriods\_BeginWorkingTimeAfterEnd\_ReturnsError()  {  var startTimes = new TimeSpan[] { };  var durations = new int[] { };  var beginWorkingTime = new TimeSpan(17, 0, 0);  var endWorkingTime = new TimeSpan(9, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "Начало рабочего дня должно быть раньше его окончания." }, result);  }  // Тест 8: Некорректные входные данные (минимальное время <= 0)  [TestMethod]  public void AvailablePeriods\_InvalidConsultationTime\_ReturnsError()  {  var startTimes = new TimeSpan[] { };  var durations = new int[] { };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 0;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "Минимальное время должно быть больше нуля." }, result);  }  // Тест 9: Некорректные входные данные (массивы разной длины)  [TestMethod]  public void AvailablePeriods\_ArraysLengthMismatch\_ReturnsError()  {  var startTimes = new[] { new TimeSpan(10, 0, 0) };  var durations = new int[] { };  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new[] { "Ошибка: массивы startTimes и durations должны быть не null и одинаковой длины." }, result);  }  // Тест 10: Занятые интервалы покрывают весь день  [TestMethod]  public void AvailablePeriods\_FullDayBusy\_ReturnsEmptyArray()  {  var startTimes = new[] { new TimeSpan(9, 0, 0) };  var durations = new[] { 480 }; // 8 часов  var beginWorkingTime = new TimeSpan(9, 0, 0);  var endWorkingTime = new TimeSpan(17, 0, 0);  var consultationTime = 30;  var result = \_calculations.AvailablePeriods(startTimes, durations, beginWorkingTime, endWorkingTime, consultationTime);  CollectionAssert.AreEqual(new string[] { }, result);  }  }  } |