МИНОБРНАУКИ РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ИЖЕВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. М.Т.Калашникова»

Кафедра «Программное обеспечение»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Исследование операций. Теория принятия решений»

на тему: «Ленточные графики. Мониторинг аренды торговых помещений в торговом центре»

Выполнил

Студент гр. Б06-191-1 Е.В. Мухачев

Проверил

Ст. преподаватель П.П. Лугачев

Ижевск 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1. ПОСТАНОВКА ЗАДАЧИ 4

2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШАЕМОЙ ЗАДАЧИ 5

3. ИНФОРМАЦИОННЫЕ СТРУКТУРЫ ДАННЫХ 7

3.1. Структура входных данных 8

3.2. Представление данных в программе 10

3.3. Структура выходных данных 10

4. СХЕМА ПРОГРАММЫ 12

4.1. Иерархическая схема программы 12

ЗАКЛЮЧЕНИЕ 15

СПИСОК ЛИТЕРАТУРЫ 16

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 17

П.1.1. Назначение программы 17

П.1.2. Условия выполнения программы 17

П.1.3. Выполнение программы 18

ПРИЛОЖЕНИЕ 2. БЛОК-СХЕМА РАБОТЫ СИСТЕМЫ 21

ПРИЛОЖЕНИЕ 3. ТЕКСТ ПРОГРАММЫ 23

ВВЕДЕНИЕ

В настоящее время остается актуальным вопрос создания программных продуктов для упрощения и автоматизации промышленных процессов и проектирования. Развивается само программирование, внедряются в повседневную жизнь все более и более сложные информационные и коммуникационные системы и технологии. Также, в последние годы четко прослеживается тенденция роста рынка коммерческой недвижимости, что и послужило основой для данной курсовой работы.

Целью данного курсового проекта является ознакомление и углубление в предмет исследований операций и теории принятия решения, а также создание программы для мониторинга аренды торгового центра, основанной на ленточном графике, на любом из объектно-ориентированных языков программирования. Актуальность проекта состоит в том, что применять данную программу можно для реального мониторинга существующего торгового центра. Так же есть возможность сохранения данных работы программы.

Программа реализована в среде разработки Microsoft Visual Studio 2017 на объектно-ориентированном языке программирования C# и VB, которые являются высокоэффективными, компактными и быстрыми языками.

1. ПОСТАНОВКА ЗАДАЧИ

Задачей курсовой работы является создание программы для мониторинга аренды помещений торгового центра.

Мониторинг аренды представляет собой карту помещения торгового центра, где показаны помещения. Каждое помещение обозначено соответствующим номером. У помещения может быть несколько арендователей, но при условии, что они арендуют помещение в разное время.

Для решения поставленной задачи необходимо использовать ленточный граф или диаграмму Гантта.

Диагра́мма Га́нтта (англ. Gantt chart, также ленточная диаграмма, график Гантта, календарный график) — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Используется в приложениях по управлению проектами.

В настоящее время диаграмма Гантта является стандартом де-факто в теории и практике управления проектами, по крайней мере, для отображения Структуры перечня работ по проекту.

1. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РЕШАЕМОЙ ЗАДАЧИ

Для решения данной задачи необходимо применить диаграмму Гантта или другой ленточный график.

Диаграмма Гантта состоит из полос, ориентированных вдоль оси времени. Каждая полоса на диаграмме представляет отдельную задачу в составе проекта (вид работы), её концы — моменты начала и завершения работы, её протяженность — длительность работы.

Вертикальной осью диаграммы служит перечень задач. Кроме того, на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи), метка текущего момента времени «Сегодня» и др.

Кроме того, диаграмма Гантта не отображает значимости или ресурсоемкости работ, не отображает сущности работ (области действия). Для крупных проектов диаграмма Гантта становится чрезмерно тяжеловесной и теряет всякую наглядность. Пример ленточной диаграммы показан на рисунке 2.1.

Ленточная диаграмма

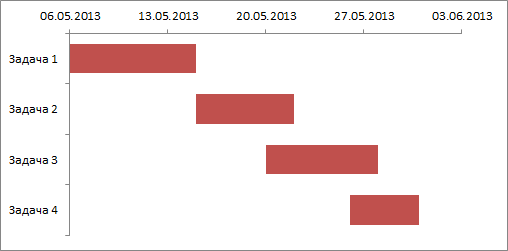


Рис. 2.1

Цена аренды помещения зависит от площади помещения и стоимости за 1кв.м. В полном варианте эта сумма должна включать в себя собственно арендную плату (ставку аренды), коммунальные платежи, налоги и эксплуатационные расходы (уборка помещения, охрана, сервисное обслуживание лифтов и пр.). Цена за аренду помещения рассчитывается по следующей формуле:

(2.1)

где Cost – цена за аренду,

Width – ширина арендуемой площади,

Height – длина арендуемой площади,

Price – цена за 1 кв. м.

Все данные по каждому помещению и арендам хранятся в отдельных файлах. При необходимости эти данные можно откорректировать.

1. ИНФОРМАЦИОННЫЕ СТРУКТУРЫ ДАННЫХ

Используемые в программе данные делятся на: входные, внутренние и выходные. Схема данных представлена на рис. 3.1.

Схема данных

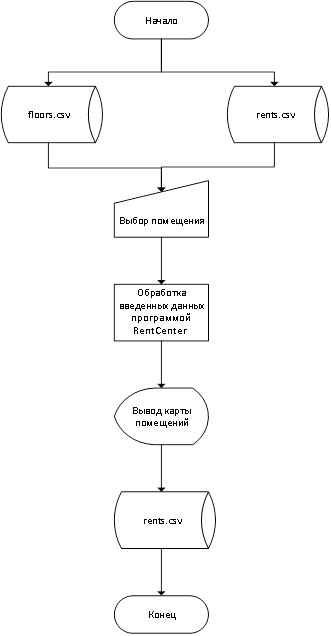


Рис. 3.1

* 1. Структура входных данных

Типизированный текстовый файл **floors.csv** — список помещений, где указано его положение на карте, ширина, длина и цена за 1 кв.м. Структура файла показана на рис. 3.2

Файл “floors.csv”



Рис. 3.2.

Типизированный текстовый файл **rents.csv** — список аренд, по каждой из аренд указывается этаж, на котором находится арендуемое помещение, номер помещения, дата начала аренды и окончания, арендователь, и цвет арендуемого помещения на карте. Структура файла показана на рис. 3.3

Файл “rents.csv”



Рис. 3.3

Пользовательские данные содержат окно приложения, где показаны все данные по арендуемому помещения. Пользователь может поменять данные по аренде каждого помещения вручную во время работы программы. Пример ввода данных представлен на рис. 3.4. Формат данных, вводимых пользователем, описан в приложении 2.

Ввод данных пользователем

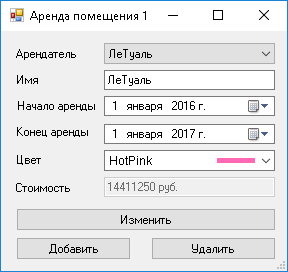


Рис. 3.4

Подробнее ввод данных пользователем описан в разделе приложении 1.3 «Выполнение программы».

* 1. Представление данных в программе

Типы:

MainWindow — класс окна приложения;

Rent — класс, содержащий информацию об аренде;

Room — класс для хранения данных помещения;

RoomDialog — класс диалогового окна изменения аренды;

GanttChart — класс ленточной диаграммы;

BarInformation — класс, использующийся для хранения данных ленточной диаграммы.

Переменные:

\_rooms — список аренд всех помещений;

\_map — карта помещений;

GanttChart — ленточная диаграмма.

* 1. Структура выходных данных

Выходными данными карта торгового центра, на которой показаны все помещения и их аренды. Также окно приложения содержит ленточный граф, показывающий время аренды по каждому арендованному помещению. Пример выходных данных представлен на рис. 3.5

Карта торгового центра



Рис. 3.5

1. СХЕМА ПРОГРАММЫ
   1. Иерархическая схема программы

Иерархическая схема программы представлена на рис. 4.1.

Иерархическая схема программы



Рис. 4.1

Спецификация:

1. **RentCenter** – решение программы, содержащее остальные проекты;
2. **RentCenter.Window** — проект основного окна приложения;
3. **RentCenter.Window.Rent –** класс, содержащий информацию по аренде помещения;
4. **RentCenter.Window.Room –** класс, содержащий информацию об помещении;
5. **RentCenter.Window.RoomDialog –** класс диалогового окна, с помощью которого происходит изменение аренды помещения;
6. **RentCenter.Window.RoomDialog.Dialog –** метод, вызывающий диалоговое окно изменение аренды;
7. **RentCenter.Window.RoomDialog.ChangeCost –** метод, изменяющий значение цены за аренду относительно выбранного периода времени;
8. **RentCenter.Window.RoomDialog.AddRent –** метод, реализующий добавление новой аренды помещения;
9. **RentCenter.Window.RoomDialog.DelRent –** метод, реализующий удаление аренды;
10. **RentCenter.Window.RoomDialog.ChangeRent –** метод, реализующий изменение текущей аренды;
11. **RentCenter.Window.RoomDialog.Select –** метод, с помощью которого происходит отображение информации по выбранной аренде;
12. **RentCenter.Window.MainWindow –** класс главного окна;
13. **RentCenter.Window.MainWindow.InitializeFloors –** метод, ответственный за загрузку данных о помещениях из файла;
14. **RentCenter.Window.MainWindow.InitializeRents –** метод, ответственный за загрузку данных об арендах из файла;
15. **RentCenter.Window.MainWindow.ChangeRent –** метод, с помощью которого происходит вызов диалогового окна изменения аренды;
16. **RentCenter.Window.MainWindow.RoomInfo –** метод, изменяющих данные на главном окне после изменения аренды;
17. **RentCenter.Window.MainWindow.FloorSelected –** метод обработки выбора этажа;
18. **RentCenter.Window.MainWindow.BarSelected –** метод обработки выбора элемента на ленточной диаграмме;
19. **RentCenter.Gantt –** проект ленточной диаграммы;
20. **RentCenter.Gantt.BarInformation –** класс, содержащий данные об элементе ленточной диаграммы;
21. **RentCenter.Gantt.GanttChart –** класс ленточной диаграммы;
22. **RentCenter.Gantt.GanttChart.AddChartBar** – метод добавления нового элемента ленточной диаграммы;
23. **RentCenter.Gantt.GanttChart.PaintChart -** метод отрисовки ленточной диаграммы;
24. **RentCenter.Gantt.GanttChart.PaintChart.DrawHeader –** метод отрисовки шапки ленточной диаграммы;
25. **RentCenter.Gantt.GanttChart.PaintChart.DrawNet -** метод отрисовки сетки ленточной диаграммы;
26. **RentCenter.Gantt.GanttChart.PaintChart.DrawBars -** метод отрисовки элементов ленточной диаграммы;
27. **RentCenter.Gantt.GanttChart.PaintChart.DrawScrollBar -** метод отрисовки скроллбара ленточной диаграммы;
28. **RentCenter.Gantt.GanttChart.GetFullHeaderList –** метод, генерирующий значения шапки;
29. **RentCenter.Gantt.GanttChart.MouseMove –**метод, ответственный за действия при наведении мыши на элемент диаграммы;
30. **RentCenter.Gantt.GanttChart.MouseMove.ToolTipDraw –** метод отрисовки подсказки при наведением мыши на элемент диаграммы;

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была реализована программа, выполняющая все выше перечисленные функции, такие как мониторинг аренды помещений торгового центра и отображение данных об арендах с помощью ленточной диаграммы. В результате использования языка C# и VB, данный программный продукт легко расширяем и модифицируем.

При выполнении работы были изучены основы теории принятия решений, ленточные диаграммы и диаграмма Гантта, их модель, определение, реализация, использованы различные математические средства, рассмотрены языки программирования C# и VB. Было сделано руководство пользователя (приложение 1), которое сможет помочь человеку ознакомиться с управлением этой программы.

СПИСОК Литературы

1. Джейсон, Прайс; Майк, Гандэрлой Visual C# .NET. Полное руководство; КОРОНА принт, 2004. - 960 c.
2. Нейгел, К. C# 2005 для профессионалов; Вильямс, 2006. - 763 c.
3. Рихтер, Джефри CLR via C#. Программирование на платформе Microsoft .NET Framework на языке C#; Питер, 2007. - 656 c.
4. Робинсон, С.; Корнес, О.; Глинн, Д. и др. C# для профессионалов; М.: Лори, 2005. - 396 c.
5. ГОСТ 19.701 – 90 (ИСО 5807 – 85) Схемы алгоритмов, программ, данных и систем. – М.: Издательство стандартов, 1991.

ПРИЛОЖЕНИЕ 1

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

П.1.1. Назначение программы

Программа предназначена для мониторинга аренды помещений торгового центра. Отображение информации осуществляется с помощью ленточной диаграммы. Учитывая возможность изменения размеров помещений, их размещения, изменения платы за 1 кв. м. и изменения данных об аренде помещения программа является очень гибкой для использования.

Выбор помещения производится на карте торгового центра. После нажатия на соответствующее помещения, программа покажет окно с текущими арендами по данному помещению. В этом окне можно изменить выбранную аренду, удалить или добавить новую.

П.1.2. Условия выполнения программы

Программа разработана для операционных систем семейства Windows версии не ниже чем Vista.

Минимальные аппаратные требования:

1. процессор с тактовой частотой 1 МГц;
2. 128 мегабайта оперативной памяти;
3. 2 МБ свободного дискового пространства.
4. Поддержка .NET Framework v4.0

Рекомендуемые аппаратные требования:

1. процессор с тактовой частотой 2 ГГц;
2. 512 мегабайт оперативной памяти;
3. более 10 МБ свободного дискового пространства.
4. Поддержка .NET Framework v4.5.2

П.1.3. Выполнение программы

Для запуска программы необходимо запустить исполняемый файл «RentCenter.exe» из каталога программы. После запуска программы открывается окно с меню картой торгового центра (рис. ПР.1.1).

Карта торгового центра



Рис. ПР.1.1

На карте пользователь может выбрать элемент диаграммы для отображения подробной информации об аренде. Пользователь также может выбрать необходимое помещение, у которого нужно изменить аренду (рис. ПР.1.2).

Выбор аренды помещения

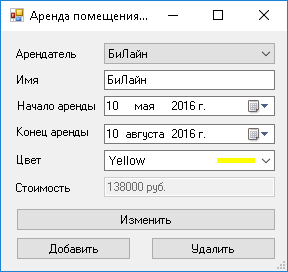


Рис. ПР.1.2

После выбора желаемых параметров аренды пользователь должен нажать соответствующую кнопку для добавления, изменения или удаления аренды. После этого в главном меню изменится информация о выбранном помещении и также изменится ленточная диаграмма. (рис. ПР.1.3).

Карта торгового центра с добавленной арендой

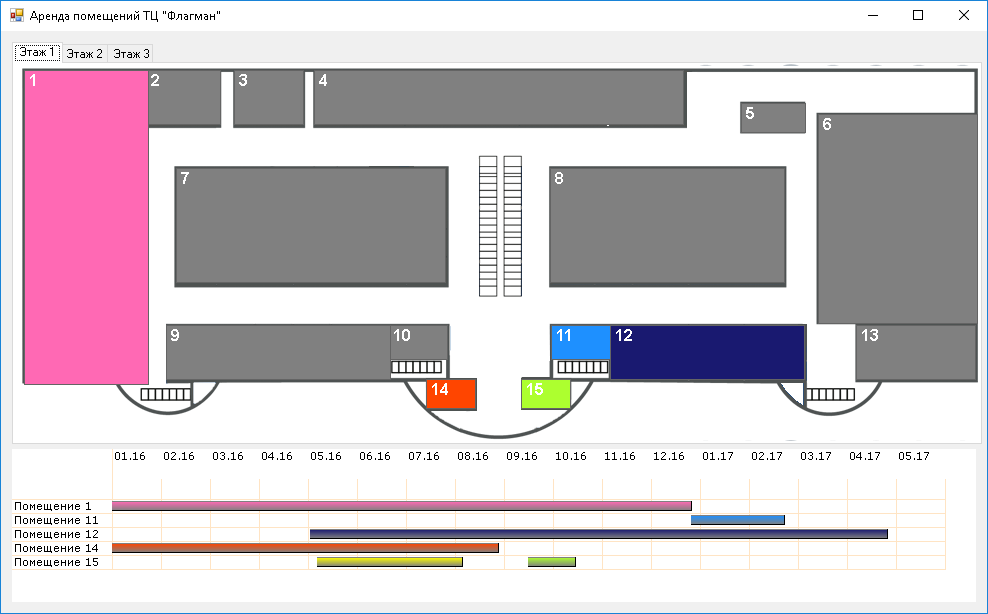


Рис. ПР.1.3

Для смены этажа торгового центра необходимо нажать на соответствующую вкладку в верхней части окна приложения. (рис. ПР.1.4).

Второй этаж торгового центра

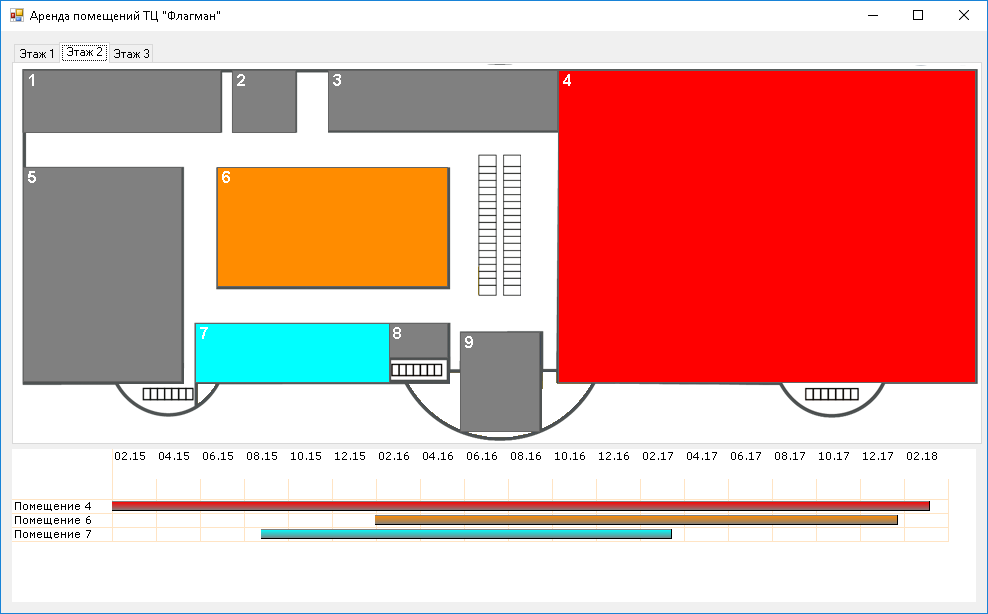


Рис. ПР.1.4

Для закрытия программы необходимо нажать на кнопку «Х», находящийся в правом верхнем углу.

Приложение 2

БЛОК-СХЕМА РАБОТЫ СИСТЕМЫ

П. 2.1. Схемы работы системы представлена на рис. ПР.2.1

Схема работы системы



Рис. ПР.2.1.

ПРИЛОЖЕНИЕ 3

ТЕКСТ ПРОГРАММЫ

using System;

using System.Windows.Forms;

namespace RentCenter.Window

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainWindow());

}

}

}

Файл “Rent.cs”

using System;

using System.Drawing;

namespace RentCenter.Window

{

public class Rent : IComparable<Rent>

{

public string Renter { get; set; }

public DateTime RentStart { get; set; }

public DateTime RentEnd { get; set; }

public Color Color { get; set; }

public Rent(string renter, DateTime start, DateTime end, Color color)

{

Renter = renter;

RentStart = start;

RentEnd = end;

Color = color;

}

public int CompareTo(Rent other)

{

if (RentEnd.CompareTo(other.RentStart) == -1) return -1;

if (RentStart.CompareTo(other.RentEnd) == 1) return 1;

return 0;

}

}

}

Файл “Room.cs”

using System.Collections.Generic;

namespace RentCenter.Window

{

public class Room

{

public bool IsArended { get; set; }

public int Price { get; }

public List<Rent> Rents { get; set; }

public int Area { get; }

public int Cost { get; }

public int Index { get; }

public int Floor { get; }

public Room(

int floor, int index,

int area, int price,

bool arend = false)

{

Rents = new List<Rent>();

Area = area;

Price = price;

Cost = area \* price;

IsArended = arend;

Floor = floor;

Index = index;

}

}

}

Файл “MainWindow.cs”

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using RentCenter.Gantt;

namespace RentCenter.Window

{

public partial class MainWindow : Form

{

private List<Room> \_rooms;

private List<PictureBox> \_map;

public MainWindow()

{

InitializeComponent();

InitializeFloors();

InitializeRents();

SelectedFloor(null,null);

FormClosing += MainWindowClosing;

}

private void MainWindowClosing(object sender, FormClosingEventArgs e)

{

var fs = new FileStream(@"..\..\rents.csv", FileMode.Create);

var sw = new StreamWriter(fs);

sw.WriteLine("Floor;RoomDialog;Rent Start;Rent End;Renter;RentColor");

foreach (var room in \_rooms)

foreach (var rent in room.Rents)

sw.WriteLine($"{room.Floor};{room.Index};{rent.RentStart};{rent.RentEnd};{rent.Renter};{rent.Color.Name}");

sw.Close();

fs.Close();

}

private void InitializeFloors()

{

var index = 1;

var lines = File.ReadAllLines(@"..\..\floors.csv");

\_map = new List<PictureBox>();

\_rooms = new List<Room>();

while (index < lines.Length)

{

var floorData = lines[index].Split(new[] {';'}, StringSplitOptions.RemoveEmptyEntries);

var floor = int.Parse(floorData[0]);

var rooms = int.Parse(floorData[1]);

index++;

for (int i = 0, room = 1; i < rooms; i++, index++, room++)

{

var roomData = lines[index].Split(new[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

var left = int.Parse(roomData[0]);

var top = int.Parse(roomData[1]);

var width = int.Parse(roomData[2]);

var height = int.Parse(roomData[3]);

var price = int.Parse(roomData[4]);

var tab = Floor1;

\_rooms.Add(new Room(floor, room, width \* height, price));

switch (floor)

{

case 1: tab = Floor1; break;

case 2: tab = Floor2; break;

case 3: tab = Floor3; break;

}

var label = new Label

{

BackColor = Color.Transparent,

ForeColor = Color.White,

Font = new Font("Arial", 12f, FontStyle.Bold),

Text = room.ToString(),

Size = new Size(30,15)

};

\_map.Add(new PictureBox

{

Parent = tab,

Size = new Size(width, height),

Top = top,

Left = left,

BorderStyle = BorderStyle.FixedSingle,

Tag = new Point(floor, room),

SizeMode = PictureBoxSizeMode.StretchImage,

BackColor = Color.Gray,

Cursor = Cursors.Hand,

});

\_map.Last().Controls.Add(label);

\_map.Last().MouseClick += RoomInfo;

}

}

}

private void InitializeRents()

{

var index = 1;

var lines = File.ReadAllLines(@"..\..\rents.csv");

while (index < lines.Length)

{

var rentData = lines[index].Split(';');

var floor = int.Parse(rentData[0]);

var room = int.Parse(rentData[1]);

var start = DateTime.Parse(rentData[2]);

var end = DateTime.Parse(rentData[3]);

var renter = rentData[4];

var color = Color.FromName(rentData[5]);

index++;

var place = \_rooms.Find(r => r.Floor == floor && r.Index == room);

place.IsArended = true;

place.Rents.Add(new Rent(renter, start, end, color));

ChangePlace(new Point(floor, room));

}

}

private void ChangePlace(Point point)

{

var place = \_map.Find(m => (Point)m.Tag == point);

var room = \_rooms.Find(r => r.Floor == point.X && r.Index == point.Y);

if (!room.IsArended) place.BackColor = Color.Gray;

else

{

var closest = room.Rents.Any(x => x.RentEnd > DateTime.Now)

? room.Rents.Where(x => x.RentEnd > DateTime.Now).OrderBy(x => x.RentEnd).First()

: room.Rents.OrderBy(x => x.RentStart).First();

place.BackColor = closest.Color;

}

place.Invalidate();

}

private void RoomInfo(object sender, MouseEventArgs e)

{

var point = (Point) (sender as Control).Tag;

var room = \_rooms.Find(r => r.Floor == point.X && r.Index == point.Y);

RoomDialog.Dialog(room);

ChangePlace(point);

SelectedFloor(null, null);

}

private void SelectedFloor(object sender, EventArgs eventArgs)

{

GanttChart.RemoveBars();

GanttChart.Invalidate();

var selected = Floors.SelectedIndex;

var floor = \_rooms.Where(r => r.IsArended && r.Floor == selected + 1).ToArray();

if(!floor.Any()) return;

GanttChart.FromDate = floor.Min(x => x.Rents.Min(y => y.RentStart));

GanttChart.ToDate = floor.Max(x => x.Rents.Max(y => y.RentEnd));

for (int i = 0; i < floor.Length; i++)

{

var room = floor[i];

foreach(var rent in room.Rents)

{

var renter = rent;

var bar = new BarInformation(

$"Помещение {room.Index}", renter.Renter,

renter.RentStart, renter.RentEnd,

renter.Color, renter.Color,

i);

GanttChart.AddChartBar(bar.RowText, bar.BarText, bar,

bar.FromTime, bar.ToTime,

bar.Color, bar.HoverColor,

bar.RowIndex);

}

}

GanttChart.Invalidate();

}

private void BarSelected(object sender, MouseEventArgs e)

{

var tips = new List<string>();

if (GanttChart.MouseOverRowText.Length > 0)

{

var data = GanttChart.MouseOverRowValue as BarInformation;

tips.Add($"Арендователь: {data.BarText}");

tips.Add($"Аренда с {data.FromTime.ToLongDateString()}");

tips.Add($"Аренда до {data.ToTime.ToLongDateString()}");

}

GanttChart.ToolTipTextTitle = GanttChart.MouseOverRowText;

GanttChart.ToolTipText = tips;

}

}

}

Файл “MainWindow.Designer.cs”

namespace RentCenter.Window

{

partial class MainWindow

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(MainWindow));

this.Floors = new System.Windows.Forms.TabControl();

this.Floor1 = new System.Windows.Forms.TabPage();

this.Floor2 = new System.Windows.Forms.TabPage();

this.Floor3 = new System.Windows.Forms.TabPage();

this.GanttChart = new RentCenter.Gantt.GanttChart();

this.Floors.SuspendLayout();

this.SuspendLayout();

this.Floors.Controls.Add(this.Floor1);

this.Floors.Controls.Add(this.Floor2);

this.Floors.Controls.Add(this.Floor3);

this.Floors.Location = new System.Drawing.Point(11, 11);

this.Floors.Margin = new System.Windows.Forms.Padding(2);

this.Floors.Name = "Floors";

this.Floors.SelectedIndex = 0;

this.Floors.Size = new System.Drawing.Size(972, 403);

this.Floors.TabIndex = 0;

this.Floors.SelectedIndexChanged += new System.EventHandler(this.SelectedFloor);

this.Floor1.BackgroundImage = global::RentCenter.Window.Properties.Resources.floor1;

this.Floor1.Location = new System.Drawing.Point(4, 22);

this.Floor1.Margin = new System.Windows.Forms.Padding(2);

this.Floor1.Name = "Floor1";

this.Floor1.Padding = new System.Windows.Forms.Padding(2);

this.Floor1.Size = new System.Drawing.Size(964, 377);

this.Floor1.TabIndex = 0;

this.Floor1.Text = "Этаж 1";

this.Floor1.UseVisualStyleBackColor = true;

this.Floor2.BackgroundImage = global::RentCenter.Window.Properties.Resources.floor2;

this.Floor2.Location = new System.Drawing.Point(4, 22);

this.Floor2.Margin = new System.Windows.Forms.Padding(2);

this.Floor2.Name = "Floor2";

this.Floor2.Padding = new System.Windows.Forms.Padding(2);

this.Floor2.Size = new System.Drawing.Size(964, 377);

this.Floor2.TabIndex = 1;

this.Floor2.Text = "Этаж 2";

this.Floor2.UseVisualStyleBackColor = true;

this.Floor3.BackgroundImage = global::RentCenter.Window.Properties.Resources.floor3;

this.Floor3.Location = new System.Drawing.Point(4, 22);

this.Floor3.Margin = new System.Windows.Forms.Padding(2);

this.Floor3.Name = "Floor3";

this.Floor3.Size = new System.Drawing.Size(964, 377);

this.Floor3.TabIndex = 2;

this.Floor3.Text = "Этаж 3";

this.Floor3.UseVisualStyleBackColor = true;

this.GanttChart.AllowManualEditBar = false;

this.GanttChart.BackColor = System.Drawing.Color.White;

this.GanttChart.DateFont = new System.Drawing.Font("Verdana", 8F);

this.GanttChart.FromDate = new System.DateTime(((long)(0)));

this.GanttChart.Location = new System.Drawing.Point(11, 418);

this.GanttChart.Margin = new System.Windows.Forms.Padding(2);

this.GanttChart.Name = "GanttChart";

this.GanttChart.RowFont = new System.Drawing.Font("Verdana", 8F);

this.GanttChart.Size = new System.Drawing.Size(964, 153);

this.GanttChart.TabIndex = 1;

this.GanttChart.Text = "GanttChart";

this.GanttChart.TimeFont = new System.Drawing.Font("Verdana", 8F);

this.GanttChart.ToDate = new System.DateTime(((long)(0)));

this.GanttChart.ToolTipText = ((System.Collections.Generic.List<string>)(resources.GetObject("GanttChart.ToolTipText")));

this.GanttChart.ToolTipTextTitle = "";

this.GanttChart.MouseMove += new System.Windows.Forms.MouseEventHandler(this.BarSelected);

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(986, 582);

this.Controls.Add(this.GanttChart);

this.Controls.Add(this.Floors);

this.Margin = new System.Windows.Forms.Padding(2);

this.Name = "MainWindow";

this.Text = "Аренда помещений ТЦ \"Флагман\"";

this.Floors.ResumeLayout(false);

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.TabControl Floors;

private System.Windows.Forms.TabPage Floor1;

private System.Windows.Forms.TabPage Floor2;

private System.Windows.Forms.TabPage Floor3;

private RentCenter.Gantt.GanttChart GanttChart;

}

}

Файл “RoomDialog.cs”

using System;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Windows.Forms;

namespace RentCenter.Window

{

public partial class RoomDialog : Form

{

private static RoomDialog \_dialogForm;

private Room \_room;

private Tuple<DateTime, DateTime> \_lastTime;

public RoomDialog()

{

InitializeComponent();

}

public static void Dialog(Room room)

{

\_dialogForm = new RoomDialog

{

\_room = room,

RentList = { DataSource = room.Rents.Select(r => r.Renter).ToArray() },

RentColor = {DataSource = typeof(Color).

GetProperties(BindingFlags.Static | BindingFlags.DeclaredOnly | BindingFlags.Public)

.Select(x=>x.Name).ToArray()}

};

\_dialogForm.Text += $" {room.Index}";

\_dialogForm.RentList.SelectedIndex = -1;

\_dialogForm.RentStart.ValueChanged += ChangeCost;

\_dialogForm.RentEnd.ValueChanged += ChangeCost;

\_dialogForm.\_lastTime = new Tuple<DateTime, DateTime>

(\_dialogForm.RentStart.MinDate, \_dialogForm.RentEnd.MaxDate);

\_dialogForm.RentColor.DrawItem += (sender, args) =>

{

var g = args.Graphics;

var rect = args.Bounds;

if (args.Index < 0) return;

var n = ((ComboBox)sender).Items[args.Index].ToString();

var f = new Font("Arial", 9, FontStyle.Regular);

var c = Color.FromName(n);

var b = new SolidBrush(c);

g.DrawString(n, f, Brushes.Black, rect.X, rect.Top);

g.FillRectangle(b, rect.X + 110, rect.Y + 5, rect.Width - 10, rect.Height - 10);

};

\_dialogForm.ShowDialog();

}

private static void ChangeCost(object sender, EventArgs e)

{

var d = \_dialogForm;

if ((d.RentEnd.Value - d.RentStart.Value).TotalDays < 1)

{

d.RentStart.Value = d.\_lastTime.Item1;

d.RentEnd.Value = d.\_lastTime.Item2;

return;

}

d.RentPrice.Text =

$@"{d.\_room.Area \* d.\_room.Price \* (d.RentEnd.Value - d.RentStart.Value).TotalDays} руб.";

d.\_lastTime = new Tuple<DateTime, DateTime>

(d.RentStart.Value, d.RentEnd.Value);

}

private void AddRent(object sender, EventArgs e)

{

if (!DataCorrect()) return;

var d = \_dialogForm;

var rent = new Rent(d.RentName.Text, d.RentStart.Value, d.RentEnd.Value,

Color.FromName((string)d.RentColor.SelectedItem));

if (d.\_room.Rents.Any(r => r.CompareTo(rent) == 0))

{

MessageBox.Show("Даты накладываются по времени", "Ошибка!", MessageBoxButtons.OK);

return;

}

\_room.Rents.Add(rent);

\_room.IsArended = true;

MessageBox.Show("Аренда добавлена", "Успешно", MessageBoxButtons.OK);

RefreshList();

}

private void DelRent(object sender, EventArgs e)

{

if (!FormCorrect()) return;

var d = \_dialogForm;

var i = d.RentList.SelectedIndex;

\_room.Rents.RemoveAt(i);

if (!\_room.Rents.Any()) \_room.IsArended = false;

d.RentList.SelectedIndex = 0;

MessageBox.Show("Аренда удалена", "Успешно", MessageBoxButtons.OK);

RefreshList();

}

private void ChangeRent(object sender, EventArgs e)

{

if (!FormCorrect() || !DataCorrect()) return;

var d = \_dialogForm;

var i = d.RentList.SelectedIndex;

var rent = new Rent(d.RentName.Text, d.RentStart.Value, d.RentEnd.Value,

Color.FromName((string) d.RentColor.SelectedItem));

if (d.\_room.Rents.Where(r => r != d.\_room.Rents[i])

.Any(r => r.CompareTo(rent) == 0))

{

MessageBox.Show("Даты накладываются по времени", "Ошибка!", MessageBoxButtons.OK);

return;

}

\_room.Rents[i].Renter = d.RentName.Text;

\_room.Rents[i].RentStart = d.RentStart.Value;

\_room.Rents[i].RentEnd = d.RentEnd.Value;

\_room.Rents[i].Color = Color.FromName((string)d.RentColor.SelectedItem);

MessageBox.Show("Аренда изменена", "Успешно", MessageBoxButtons.OK);

RefreshList();

}

private void SelectRent(object sender, EventArgs e)

{

if(!FormCorrect()) return;

var d = \_dialogForm;

var i = d.RentList.SelectedIndex;

d.RentColor.SelectedItem = \_room.Rents[i].Color.Name;

d.RentName.Text = d.RentList.SelectedItem.ToString();

d.RentStart.Value = \_room.Rents[i].RentStart;

d.RentEnd.Value = \_room.Rents[i].RentEnd;

d.\_lastTime = new Tuple<DateTime, DateTime>

(d.RentStart.Value, d.RentEnd.Value);

}

private void RefreshList()

{

\_dialogForm.RentList.DataSource = \_room.Rents.Select(r => r.Renter).ToArray();

}

private static bool FormCorrect()

{

if (\_dialogForm == null) return false;

return \_dialogForm.RentList.SelectedIndex != -1;

}

private static bool DataCorrect()

{

if (string.IsNullOrEmpty(\_dialogForm.RentName.Text)) return false;

return \_dialogForm.RentColor.Text != "Transparent";

}

}

}

Файл “RoomDialog.Designer.cs”

namespace RentCenter.Window

{

partial class RoomDialog

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

private void InitializeComponent()

{

this.RentList = new System.Windows.Forms.ComboBox();

this.label1 = new System.Windows.Forms.Label();

this.RentStart = new System.Windows.Forms.DateTimePicker();

this.RentEnd = new System.Windows.Forms.DateTimePicker();

this.RentColor = new System.Windows.Forms.ComboBox();

this.RentName = new System.Windows.Forms.TextBox();

this.AddButton = new System.Windows.Forms.Button();

this.DelButton = new System.Windows.Forms.Button();

this.RentPrice = new System.Windows.Forms.TextBox();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.label6 = new System.Windows.Forms.Label();

this.ChangeButton = new System.Windows.Forms.Button();

this.SuspendLayout();

this.RentList.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

this.RentList.FormattingEnabled = true;

this.RentList.Location = new System.Drawing.Point(103, 12);

this.RentList.Name = "RentList";

this.RentList.Size = new System.Drawing.Size(171, 21);

this.RentList.TabIndex = 0;

this.RentList.SelectedIndexChanged += new System.EventHandler(this.SelectRent);

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(12, 16);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(67, 13);

this.label1.TabIndex = 1;

this.label1.Text = "Арендатель";

this.RentStart.Location = new System.Drawing.Point(103, 65);

this.RentStart.MaxDate = new System.DateTime(2030, 12, 31, 0, 0, 0, 0);

this.RentStart.MinDate = new System.DateTime(1990, 1, 1, 0, 0, 0, 0);

this.RentStart.Name = "RentStart";

this.RentStart.Size = new System.Drawing.Size(171, 20);

this.RentStart.TabIndex = 2;

this.RentStart.Value = new System.DateTime(1990, 1, 1, 0, 0, 0, 0);

this.RentEnd.Location = new System.Drawing.Point(103, 93);

this.RentEnd.MaxDate = new System.DateTime(2030, 12, 31, 0, 0, 0, 0);

this.RentEnd.MinDate = new System.DateTime(1990, 1, 1, 0, 0, 0, 0);

this.RentEnd.Name = "RentEnd";

this.RentEnd.Size = new System.Drawing.Size(171, 20);

this.RentEnd.TabIndex = 3;

this.RentEnd.Value = new System.DateTime(2030, 12, 31, 0, 0, 0, 0);

this.RentColor.DrawMode = System.Windows.Forms.DrawMode.OwnerDrawVariable;

this.RentColor.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

this.RentColor.FormattingEnabled = true;

this.RentColor.Location = new System.Drawing.Point(103, 119);

this.RentColor.Name = "RentColor";

this.RentColor.Size = new System.Drawing.Size(171, 21);

this.RentColor.TabIndex = 4;

this.RentName.Location = new System.Drawing.Point(103, 39);

this.RentName.Name = "RentName";

this.RentName.Size = new System.Drawing.Size(171, 20);

this.RentName.TabIndex = 5;

this.AddButton.Location = new System.Drawing.Point(15, 206);

this.AddButton.Name = "AddButton";

this.AddButton.Size = new System.Drawing.Size(115, 23);

this.AddButton.TabIndex = 6;

this.AddButton.Text = "Добавить";

this.AddButton.UseVisualStyleBackColor = true;

this.AddButton.Click += new System.EventHandler(this.AddRent);

this.DelButton.Location = new System.Drawing.Point(150, 206);

this.DelButton.Name = "DelButton";

this.DelButton.Size = new System.Drawing.Size(125, 23);

this.DelButton.TabIndex = 7;

this.DelButton.Text = "Удалить";

this.DelButton.UseVisualStyleBackColor = true;

this.DelButton.Click += new System.EventHandler(this.DelRent);

this.RentPrice.Enabled = false;

this.RentPrice.Location = new System.Drawing.Point(103, 146);

this.RentPrice.Name = "RentPrice";

this.RentPrice.Size = new System.Drawing.Size(171, 20);

this.RentPrice.TabIndex = 8;

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(12, 42);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(29, 13);

this.label2.TabIndex = 9;

this.label2.Text = "Имя";

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(13, 68);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(85, 13);

this.label3.TabIndex = 10;

this.label3.Text = "Начало аренды";

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(12, 94);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(79, 13);

this.label4.TabIndex = 11;

this.label4.Text = "Конец аренды";

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(12, 122);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(32, 13);

this.label5.TabIndex = 12;

this.label5.Text = "Цвет";

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(11, 150);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(62, 13);

this.label6.TabIndex = 13;

this.label6.Text = "Стоимость";

this.ChangeButton.Location = new System.Drawing.Point(15, 177);

this.ChangeButton.Name = "ChangeButton";

this.ChangeButton.Size = new System.Drawing.Size(260, 23);

this.ChangeButton.TabIndex = 14;

this.ChangeButton.Text = "Изменить";

this.ChangeButton.UseVisualStyleBackColor = true;

this.ChangeButton.Click += new System.EventHandler(this.ChangeRent);

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(286, 240);

this.Controls.Add(this.ChangeButton);

this.Controls.Add(this.label6);

this.Controls.Add(this.label5);

this.Controls.Add(this.label4);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.RentPrice);

this.Controls.Add(this.DelButton);

this.Controls.Add(this.AddButton);

this.Controls.Add(this.RentName);

this.Controls.Add(this.RentColor);

this.Controls.Add(this.RentEnd);

this.Controls.Add(this.RentStart);

this.Controls.Add(this.label1);

this.Controls.Add(this.RentList);

this.Name = "RoomDialog";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;

this.Text = "Аренда помещения";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.ComboBox RentList;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.DateTimePicker RentStart;

private System.Windows.Forms.DateTimePicker RentEnd;

private System.Windows.Forms.ComboBox RentColor;

private System.Windows.Forms.TextBox RentName;

private System.Windows.Forms.Button AddButton;

private System.Windows.Forms.Button DelButton;

private System.Windows.Forms.TextBox RentPrice;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.Button ChangeButton;

}

}

Файл “BarInformation.vb”

Public Class BarInformation

Public Property BarText As String

Public Property RowText As String

Public Property FromTime As Date

Public Property ToTime As Date

Public Property Color As Color

Public Property HoverColor As Color

Public Property RowIndex As Integer

Public Sub New(rowText As String, barText As String, fromTime As Date, totime As Date, color As Color, hoverColor As Color, ByVal rowIndex As Integer)

Me.RowText = rowText

Me.BarText = barText

Me.FromTime = fromTime

Me.ToTime = totime

Me.Color = color

Me.HoverColor = hoverColor

Me.RowIndex = rowIndex

End Sub

End Class

Файл “GanttChart.vb”

Imports System.Drawing.Drawing2D

Public Class GanttChart

Inherits Control

Private \_mouseHoverPart As MouseOverPart = MouseOverPart.Empty

Private \_mouseHoverBarIndex As Integer = -1

Private \_bars As New List(Of ChartBarDate)

Private \_headerFromDate As Date = Nothing

Private \_headerToDate As Date = Nothing

Private \_barIsChanging As Integer = -1

Private \_barStartRight As Integer = 20

Private \_barStartLeft As Integer = 100

Private \_headerTimeStartTop As Integer = 30

Private \_shownHeaderList As List(Of Header)

Private \_barStartTop As Integer = 50

Private \_barHeight As Integer = 9

Private \_barSpace As Integer = 5

Private \_widthPerItem As Integer

Private \_mouseOverColumnValue As Date = Nothing

Private \_mouseOverRowText As String = ""

Private \_mouseOverRowValue As Object = Nothing

Private \_lineColor As Pen = Pens.Bisque

Private \_dateTextFont As Font = New Font("VERDANA", 8.0, FontStyle.Regular, GraphicsUnit.Point)

Private \_timeTextFont As Font = New Font("VERDANA", 8.0, FontStyle.Regular, GraphicsUnit.Point)

Private \_rowTextFont As Font = New Font("VERDANA", 8.0, FontStyle.Regular, GraphicsUnit.Point)

Friend WithEvents ToolTip As New ToolTip()

Public Event MouseDragged(sender As Object,e As MouseEventArgs)

Public Event BarChanged(sender As Object, ByRef barValue As Object)

Private \_objBmp As Bitmap

Private \_objGraphics As Graphics

#Region "Public properties"

Public Property AllowManualEditBar As Boolean = False

Public Property FromDate As Date

Get

Return \_headerFromDate

End Get

Set

\_headerFromDate = value

End Set

End Property

Public Property ToDate As Date

Get

Return \_headerToDate

End Get

Set

\_headerToDate = value

End Set

End Property

Public ReadOnly Property MouseOverRowText As String

Get

Return \_mouseOverRowText

End Get

End Property

Public ReadOnly Property MouseOverRowValue As Object

Get

Return \_mouseOverRowValue

End Get

End Property

Public ReadOnly Property MouseOverColumnDate As Date

Get

Return \_mouseOverColumnValue

End Get

End Property

Public Property GridColor As Pen

Get

Return \_lineColor

End Get

Set

\_lineColor = value

End Set

End Property

Public Property RowFont As Font

Get

Return \_rowTextFont

End Get

Set

\_rowTextFont = value

End Set

End Property

Public Property DateFont As Font

Get

Return \_dateTextFont

End Get

Set

\_dateTextFont = value

End Set

End Property

Public Property TimeFont As Font

Get

Return \_timeTextFont

End Get

Set

\_timeTextFont = value

End Set

End Property

#End Region

#Region "Constructor"

Public Sub New()

ToolTip.AutoPopDelay = 15000

ToolTip.InitialDelay = 250

ToolTip.OwnerDraw = True

\_objBmp = New Bitmap(1280, 1024, Imaging.PixelFormat.Format24bppRgb)

\_objGraphics = Graphics.FromImage(\_objBmp)

SetStyle(ControlStyles.DoubleBuffer Or ControlStyles.UserPaint Or ControlStyles.AllPaintingInWmPaint, True)

End Sub

#End Region

#Region "Bars"

Private Sub SetBarStartLeft(rowText As String)

Dim gfx As Graphics = CreateGraphics

Dim length As Integer = gfx.MeasureString(rowText, \_rowTextFont, 500).Width

If length > \_barStartLeft Then

\_barStartLeft = length

End If

End Sub

Public Sub AddChartBar(rowText As String, barText As String, barValue As Object, fromTime As Date, toTime As Date, color As Color, hoverColor As Color, rowIndex As Integer)

Dim bar As New ChartBarDate

bar.Name = barText

bar.Text = rowText

bar.Value = barValue

bar.StartValue = fromTime

bar.EndValue = toTime

bar.Color = color

bar.HoverColor = hoverColor

bar.RowIndex = rowIndex

\_bars.Add(bar)

SetBarStartLeft(rowText)

End Sub

Public Sub AddChartBar(rowText As String, barText As String, barValue As Object, fromTime As Date, toTime As Date, color As Color, hoverColor As Color, rowIndex As Integer, hideFromMouseMove As Boolean)

Dim bar As New ChartBarDate

bar.Name = barText

bar.Text = rowText

bar.Value = barValue

bar.StartValue = fromTime

bar.EndValue = toTime

bar.Color = color

bar.HoverColor = hoverColor

bar.RowIndex = rowIndex

bar.HideFromMouseMove = hideFromMouseMove

\_bars.Add(bar)

SetBarStartLeft(rowText)

End Sub

Public Function GetIndexChartBar(rowText As String) As Integer

Dim index As Integer = -1

For Each bar As ChartBarDate In \_bars

If bar.Text.Equals(rowText) = True Then

Return bar.RowIndex

End If

If bar.RowIndex > index Then

index = bar.RowIndex

End If

Next

Return index + 1

End Function

Public Sub RemoveBars()

\_bars = New List(Of ChartBarDate)

\_barStartLeft = 100

End Sub

#End Region

#Region "Draw"

Public Sub PaintChart()

Invalidate()

End Sub

Private Sub PaintChart(gfx As Graphics)

gfx.Clear(BackColor)

If \_headerFromDate = Nothing Or \_headerToDate = Nothing Then Exit Sub

DrawScrollBar(gfx)

DrawHeader(gfx, Nothing)

DrawNetHorizontal(gfx)

DrawNetVertical(gfx)

DrawBars(gfx)

\_objBmp = New Bitmap(Width - \_barStartRight, \_lastLineStop, Imaging.PixelFormat.Format24bppRgb)

\_objGraphics = Graphics.FromImage(\_objBmp)

End Sub

Protected Overrides Sub OnPaint(pe As PaintEventArgs)

MyBase.OnPaint(pe)

PaintChart(pe.Graphics)

End Sub

Private Sub DrawHeader(gfx As Graphics, headerList As List(Of Header))

If headerList Is Nothing Then

headerList = GetFullHeaderList()

End If

If headerList.Count = 0 Then Exit Sub

Dim availableWidth = Width - 10 - \_barStartLeft - \_barStartRight

\_widthPerItem = availableWidth / headerList.Count

If \_widthPerItem < 40 Then

Dim newHeaderList As New List(Of Header)

Dim showNext As Boolean = True

For Each header As Header In headerList

If showNext = True Then

newHeaderList.Add(header)

showNext = False

Else

showNext = True

End If

Next

DrawHeader(gfx, newHeaderList)

Exit Sub

End If

Dim index As Integer = 0

Dim lastHeader As Header = Nothing

For Each header As Header In headerList

Dim startPos As Integer = \_barStartLeft + (index \* \_widthPerItem)

Dim showDateHeader As Boolean = False

header.StartLocation = startPos

If lastHeader Is Nothing Then

showDateHeader = True

ElseIf header.Time.Hour < lastHeader.Time.Hour Then

showDateHeader = True

ElseIf header.Time.Minute = lastHeader.Time.Minute Then

showDateHeader = True

End If

If showDateHeader = True Then

Dim str As String

If header.HeaderTextInsteadOfTime.Length > 0 Then

str = header.HeaderTextInsteadOfTime

Else

str = header.Time.ToString("d-MMM")

End If

gfx.DrawString(str, \_dateTextFont, Brushes.Black, startPos, 0)

End If

gfx.DrawString(header.HeaderText, \_timeTextFont, Brushes.Black, startPos, \_headerTimeStartTop)

index += 1

lastHeader = header

Next

\_shownHeaderList = headerList

\_widthPerItem = (Width - 10 - \_barStartLeft - \_barStartRight) / \_shownHeaderList.Count

End Sub

Private Sub DrawBars(grfx As Graphics, Optional ByVal ignoreScrollAndMousePosition As Boolean = False)

If \_shownHeaderList Is Nothing Then Exit Sub

If \_shownHeaderList.Count = 0 Then Exit Sub

Dim index As Integer

Dim timeBetween As TimeSpan = \_shownHeaderList(1).Time - \_shownHeaderList(0).Time

Dim minutesBetween As Integer = (timeBetween.TotalMinutes)

Dim widthBetween = (\_shownHeaderList(1).StartLocation - \_shownHeaderList(0).StartLocation)

Dim perMinute As Decimal = widthBetween / minutesBetween

For Each bar As ChartBarDate In \_bars

index = bar.RowIndex

Dim startLocation As Integer

Dim startMinutes As Integer

Dim startTimeSpan As TimeSpan

Dim lengthMinutes As Integer

Dim scrollPos As Integer = 0

If ignoreScrollAndMousePosition = False Then

scrollPos = \_scrollPosition

End If

startTimeSpan = bar.StartValue - FromDate

startMinutes = (startTimeSpan.Days \* 1440) + (startTimeSpan.Hours \* 60) + startTimeSpan.Minutes

startLocation = perMinute \* startMinutes

Dim endValue As Date = bar.EndValue

If endValue = Nothing Then

endValue = Date.Now

End If

Dim lengthTimeSpan As TimeSpan = endValue - bar.StartValue

lengthMinutes = (lengthTimeSpan.Days \* 1440) + (lengthTimeSpan.Hours \* 60) + lengthTimeSpan.Minutes

Dim widths As Integer = perMinute \* lengthMinutes

Dim a As Integer = \_barStartLeft + startLocation

Dim b As Integer = \_barStartTop + (\_barHeight \* (index - scrollPos)) + (\_barSpace \* (index - scrollPos)) + 2

Dim c As Integer = widths

Dim d As Integer = \_barHeight

If c = 0 Then c = 1

If a - \_barStartLeft < 0 Then

a = \_barStartLeft

End If

Dim color As Color

If MouseOverRowText = bar.Text And bar.StartValue <= \_mouseOverColumnValue And bar.EndValue >= \_mouseOverColumnValue Then

color = bar.HoverColor

Else

color = bar.Color

End If

bar.TopLocation.Left = New Point(a, b)

bar.TopLocation.Right = New Point(a + c, b)

bar.BottomLocation.Left = New Point(a, b + d)

bar.BottomLocation.Right = New Point(a, b + d)

Dim obBrush As LinearGradientBrush

Dim obRect As New Rectangle(a, b, c, d)

If bar.StartValue <> Nothing And endValue <> Nothing Then

If (index >= scrollPos And index < \_barsViewable + scrollPos) Or ignoreScrollAndMousePosition = True Then

obBrush = New LinearGradientBrush(obRect, color, color.Gray, LinearGradientMode.Vertical)

grfx.DrawRectangle(Pens.Black, obRect)

grfx.FillRectangle(obBrush, obRect)

grfx.DrawString(bar.Text, \_rowTextFont, Brushes.Black, 0, \_barStartTop + (\_barHeight \* (index - scrollPos)) + (\_barSpace \* (index - scrollPos)))

End If

End If

Next

End Sub

Public Sub DrawNetVertical(grfx As Graphics)

If \_shownHeaderList Is Nothing Then Exit Sub

If \_shownHeaderList.Count = 0 Then Exit Sub

Dim index As Integer = 0

Dim lastHeader As Header = Nothing

For Each header As Header In \_shownHeaderList

Dim headerLocationY As Integer

If lastHeader Is Nothing Then

headerLocationY = 0

ElseIf header.Time.Hour < lastHeader.Time.Hour Then

headerLocationY = 0

Else

headerLocationY = \_headerTimeStartTop

End If

grfx.DrawLine(Pens.Bisque, \_barStartLeft + (index \* \_widthPerItem), headerLocationY, \_barStartLeft + (index \* \_widthPerItem), \_lastLineStop)

index += 1

lastHeader = header

Next

grfx.DrawLine(\_lineColor, \_barStartLeft + (index \* \_widthPerItem), \_headerTimeStartTop, \_barStartLeft + (index \* \_widthPerItem), \_lastLineStop)

End Sub

Public Sub DrawNetHorizontal(grfx As Graphics)

If \_shownHeaderList Is Nothing Then Exit Sub

If \_shownHeaderList.Count = 0 Then Exit Sub

Dim index As Integer

Dim widths As Integer = (\_widthPerItem \* \_shownHeaderList.Count) + \_barStartLeft

For index = 0 To GetIndexChartBar("QQQQQQ")

For Each bar As ChartBarDate In \_bars

grfx.DrawLine(\_lineColor, 0, \_barStartTop + (\_barHeight \* index) + (\_barSpace \* index), widths, \_barStartTop + (\_barHeight \* index) + (\_barSpace \* index))

Next

Next

\_lastLineStop = \_barStartTop + (\_barHeight \* (index - 1)) + (\_barSpace \* (index - 1))

End Sub

Private \_lastLineStop As Integer = 0

#End Region

#Region "Header list"

Private Function GetFullHeaderList() As List(Of Header)

Dim result As New List(Of Header)

Dim newFromTime As New Date(FromDate.Year, FromDate.Month, FromDate.Day)

Dim item As String

Dim interval As TimeSpan = ToDate - FromDate

If interval.TotalDays < 1 Then

With newFromTime

newFromTime = .AddHours(FromDate.Hour)

If \_headerFromDate.Minute < 59 And \_headerFromDate.Minute > 29 Then

newFromTime = .AddMinutes(30)

Else

newFromTime = .AddMinutes(0)

End If

End With

While newFromTime <= ToDate

item = newFromTime.Hour & ":"

If newFromTime.Minute < 10 Then

item += "0" & newFromTime.Minute

Else

item += "" & newFromTime.Minute

End If

Dim header As New Header

header.HeaderText = item

header.HeaderTextInsteadOfTime = ""

header.Time = New Date(newFromTime.Year, newFromTime.Month, newFromTime.Day, newFromTime.Hour, newFromTime.Minute, 0)

result.Add(header)

newFromTime = newFromTime.AddMinutes(5)

End While

ElseIf interval.TotalDays < 60 Then

While newFromTime <= ToDate

Dim header As New Header

header.HeaderText = ""

header.HeaderTextInsteadOfTime = ""

header.Time = New Date(newFromTime.Year, newFromTime.Month, newFromTime.Day, 0, 0, 0)

result.Add(header)

newFromTime = newFromTime.AddDays(1)

End While

Else

While newFromTime <= ToDate

Dim header As New Header

header.HeaderText = ""

header.Time = New Date(newFromTime.Year, newFromTime.Month, newFromTime.Day, 0, 0, 0)

header.HeaderTextInsteadOfTime = newFromTime.ToString("MM.yy")

result.Add(header)

newFromTime = newFromTime.AddMonths(1)

End While

End If

Return result

End Function

#End Region

#Region "Mouse Move"

Private Sub GanttChart\_MouseMove(sender As System.Object, e As MouseEventArgs) Handles MyBase.MouseMove

If \_shownHeaderList Is Nothing Then Exit Sub

If \_shownHeaderList.Count = 0 Then Exit Sub

If e.Button <> MouseButtons.Left Then

\_mouseHoverPart = MouseOverPart.Empty

If AllowManualEditBar = True Then

If \_barIsChanging >= 0 Then

RaiseEvent BarChanged(Me, \_bars(\_barIsChanging).Value)

\_barIsChanging = -1

End If

End If

End If

\_mouseHoverBarIndex = -1

Dim localMousePosition As Point

localMousePosition = PointToClient(Cursor.Position)

Dim timeBetween As TimeSpan = \_shownHeaderList(1).Time - \_shownHeaderList(0).Time

Dim minutesBetween As Integer = (timeBetween.Days \* 1440) + (timeBetween.Hours \* 60) + timeBetween.Minutes

Dim widthBetween = (\_shownHeaderList(1).StartLocation - \_shownHeaderList(0).StartLocation)

Dim perMinute As Decimal = widthBetween / minutesBetween

Dim minutesAtCursor As Integer

If localMousePosition.X > \_barStartLeft Then

minutesAtCursor = (localMousePosition.X - \_barStartLeft) / perMinute

\_mouseOverColumnValue = FromDate.AddMinutes(minutesAtCursor)

Else

\_mouseOverColumnValue = Nothing

End If

Dim rowText As String = ""

Dim rowValue As Object = Nothing

Dim scrollBarStatusChanged As Boolean = False

If localMousePosition.X > \_bottomPart.Left And localMousePosition.Y < \_bottomPart.Right And localMousePosition.Y < \_bottomPart.Bottom And localMousePosition.Y > \_bottomPart.Top Then

If \_mouseOverBottomPart = False Then

scrollBarStatusChanged = True

End If

\_mouseOverBottomPart = True

Else

If \_mouseOverBottomPart = False Then

scrollBarStatusChanged = True

End If

\_mouseOverBottomPart = False

End If

If localMousePosition.X > \_topPart.Left And localMousePosition.Y < \_topPart.Right And localMousePosition.Y < \_topPart.Bottom And localMousePosition.Y > \_topPart.Top Then

If \_mouseOverTopPart = False Then

scrollBarStatusChanged = True

End If

\_mouseOverTopPart = True

Else

If \_mouseOverTopPart = False Then

scrollBarStatusChanged = True

End If

\_mouseOverTopPart = False

End If

If localMousePosition.X > \_scroll.Left And localMousePosition.Y < \_scroll.Right And localMousePosition.Y < \_scroll.Bottom And localMousePosition.Y > \_scroll.Top Then

If \_mouseOverScrollBar = False Then

scrollBarStatusChanged = True

End If

\_mouseOverScrollBar = True

\_mouseOverScrollBarArea = True

Else

If \_mouseOverScrollBar = False Then

scrollBarStatusChanged = True

End If

\_mouseOverScrollBar = False

\_mouseOverScrollBarArea = False

End If

If \_mouseOverScrollBarArea = False Then

If localMousePosition.X > \_scrollBarArea.Left And localMousePosition.Y < \_scrollBarArea.Right And localMousePosition.Y < \_scrollBarArea.Bottom And localMousePosition.Y > \_scrollBarArea.Top Then

\_mouseOverScrollBarArea = True

End If

End If

Dim index As Integer = 0

For Each bar As ChartBarDate In \_bars

If bar.HideFromMouseMove = False Then

If bar.EndValue = Nothing Then

bar.EndValue = Date.Now

End If

If localMousePosition.Y > bar.TopLocation.Left.Y And localMousePosition.Y < bar.BottomLocation.Left.Y Then

If localMousePosition.X > bar.TopLocation.Left.X And localMousePosition.X < bar.TopLocation.Right.X Then

rowText = bar.Text

rowValue = bar.Value

\_mouseHoverBarIndex = index

If \_mouseHoverPart <> MouseOverPart.BarLeftSide And \_mouseHoverPart <> MouseOverPart.BarRightSide Then

\_mouseHoverPart = MouseOverPart.Bar

End If

End If

If AllowManualEditBar = True Then

Dim areaSize As Integer = 5

If e.Button = Windows.Forms.MouseButtons.Left Then

areaSize = 50

End If

If localMousePosition.X > bar.TopLocation.Left.X - areaSize And localMousePosition.X < bar.TopLocation.Left.X + areaSize And \_mouseHoverPart <> MouseOverPart.BarRightSide Then

Me.Cursor = Cursors.VSplit

\_mouseHoverPart = MouseOverPart.BarLeftSide

\_mouseHoverBarIndex = index

ElseIf localMousePosition.X > bar.TopLocation.Right.X - areaSize And localMousePosition.X < bar.TopLocation.Right.X + areaSize And \_mouseHoverPart <> MouseOverPart.BarLeftSide Then

Me.Cursor = Cursors.VSplit

\_mouseHoverPart = MouseOverPart.BarRightSide

\_mouseHoverBarIndex = index

Else

Me.Cursor = Cursors.Default

End If

End If

End If

End If

index += 1

Next

\_mouseOverRowText = rowText

\_mouseOverRowValue = rowValue

If e.Button = Windows.Forms.MouseButtons.Left Then

RaiseEvent MouseDragged(sender, e)

Else

If (\_mouseOverRowValue Is Nothing And Not rowValue Is Nothing) Or (Not \_mouseOverRowValue Is Nothing And rowValue Is Nothing) Or scrollBarStatusChanged = True Then

PaintChart()

End If

End If

End Sub

Private Sub GanttChart\_MouseLeave(sender As System.Object, e As EventArgs) Handles MyBase.MouseLeave

\_mouseOverRowText = Nothing

\_mouseOverRowValue = Nothing

\_mouseHoverPart = MouseOverPart.Empty

PaintChart()

End Sub

Public Sub GanttChart\_MouseDragged(sender As Object, e As MouseEventArgs) Handles Me.MouseDragged

If \_mouseOverScrollBarArea = True Then

ScrollPositionY = e.Location.Y

End If

If AllowManualEditBar = True Then

If \_mouseHoverBarIndex > -1 Then

If \_mouseHoverPart = MouseOverPart.BarLeftSide Then

\_barIsChanging = \_mouseHoverBarIndex

\_bars(\_mouseHoverBarIndex).StartValue = \_mouseOverColumnValue

PaintChart()

ElseIf \_mouseHoverPart = MouseOverPart.BarRightSide Then

\_barIsChanging = \_mouseHoverBarIndex

\_bars(\_mouseHoverBarIndex).EndValue = \_mouseOverColumnValue

PaintChart()

End If

End If

End If

End Sub

#End Region

#Region "ToolTipText"

Private \_toolTipText As New List(Of String)

Private \_myPoint As New Point(0, 0)

Public Property ToolTipTextTitle As String = ""

Public Property ToolTipText() As List(Of String)

Get

If \_toolTipText Is Nothing Then \_toolTipText = New List(Of String)

Return \_toolTipText

End Get

Set

\_toolTipText = value

Dim localMousePosition As Point

localMousePosition = Me.PointToClient(Cursor.Position)

If localMousePosition = \_myPoint Then Exit Property

\_myPoint = localMousePosition

ToolTip.SetToolTip(Me, ".")

End Set

End Property

Private Sub ToolTipText\_Draw(sender As System.Object, e As DrawToolTipEventArgs) Handles ToolTip.Draw

If ToolTipText Is Nothing Then

ToolTipText = New List(Of String)

Exit Sub

End If

If ToolTipText.Count = 0 Then

Exit Sub

ElseIf ToolTipText(0).Length = 0 Then

Exit Sub

End If

Dim x As Integer

Dim y As Integer

e.Graphics.FillRectangle(Brushes.AntiqueWhite, e.Bounds)

e.DrawBorder()

Dim titleHeight As Integer = 14

Dim fontHeight As Integer = 12

e.Graphics.DrawLine(Pens.Black, 0, titleHeight, e.Bounds.Width, titleHeight)

Dim lines As Integer = 1

Dim texts As String = ToolTipTextTitle

Using fonts As New Font(e.Font, FontStyle.Bold)

x = (e.Bounds.Width - e.Graphics.MeasureString(texts, fonts).Width) \ 2

y = (titleHeight - e.Graphics.MeasureString(texts, fonts).Height) \ 2

e.Graphics.DrawString(texts, fonts, Brushes.Black, x, y)

End Using

For Each str As String In ToolTipText

Dim fonts As New Font(e.Font, FontStyle.Regular)

If str.Contains("[b]") Then

fonts = New Font(fonts.FontFamily, fonts.Size, FontStyle.Bold, fonts.Unit)

str = str.Replace("[b]", "")

End If

Using fonts

x = 5

y = (titleHeight - fontHeight - e.Graphics.MeasureString(str, fonts).Height) \ 2 + 10 + (lines \* 14)

e.Graphics.DrawString(str, fonts, Brushes.Black, x, y)

End Using

lines += 1

Next

End Sub

Private Sub ToolTipText\_Popup(sender As System.Object, e As PopupEventArgs) Handles ToolTip.Popup

If ToolTipText Is Nothing Then

ToolTipText = New List(Of String)

End If

If ToolTipText.Count = 0 Then

e.ToolTipSize = New Size(0, 0)

Exit Sub

ElseIf ToolTipText(0).Length = 0 Then

e.ToolTipSize = New Size(0, 0)

Exit Sub

End If

Dim heights As Integer = 18 + (ToolTipText.Count \* 15)

e.ToolTipSize = New Size(200, heights)

End Sub

#End Region

#Region "ChartBar"

Public Class ChartBarDate

Friend Class Location

Public Property Right As Point = New Point(0, 0)

Public Property Left As Point = New Point(0, 0)

End Class

Public Property StartValue As Date

Public Property EndValue As Date

Public Property Color As Color

Public Property HoverColor As Color

Public Property Name As String

Public Property Text As String

Public Property Value As Object

Public Property RowIndex As Integer

Public Property HideFromMouseMove As Boolean = False

Friend Property TopLocation As Location = New Location

Friend Property BottomLocation As Location = New Location

End Class

#End Region

#Region "Headers"

Private Class Header

Public Property HeaderText As String

Public Property StartLocation As Integer

Public Property HeaderTextInsteadOfTime As String = ""

Public Property Time As Date = Nothing

End Class

#End Region

#Region "Resize"

Protected Overrides Sub OnResize(e As EventArgs)

MyBase.OnResize(e)

\_scrollPosition = 0

If \_lastLineStop > 0 Then

\_objBmp = New Bitmap(Width - \_barStartRight, \_lastLineStop, Imaging.PixelFormat.Format24bppRgb)

\_objGraphics = Graphics.FromImage(\_objBmp)

End If

PaintChart()

End Sub

#End Region

#Region "Scrollbar"

Private \_barsViewable As Integer = -1

Private \_scrollPosition As Integer = 0

Private \_topPart As Rectangle = Nothing

Private \_bottomPart As Rectangle = Nothing

Private \_scroll As Rectangle = Nothing

Private \_scrollBarArea As Rectangle = Nothing

Private \_mouseOverTopPart As Boolean = False

Private \_mouseOverBottomPart As Boolean = False

Private \_mouseOverScrollBar As Boolean = False

Private \_mouseOverScrollBarArea As Boolean = False

Private Sub DrawScrollBar(grfx As Graphics)

\_barsViewable = (Height - \_barStartTop) / (\_barHeight + \_barSpace)

Dim barCount As Integer = GetIndexChartBar("QQQWWW")

If barCount = 0 Then Exit Sub

Dim maxHeight As Integer = Height - 30

Dim scrollHeight As Decimal = (maxHeight / barCount) \* \_barsViewable

If scrollHeight >= maxHeight Then Exit Sub

Dim scrollSpeed As Decimal = (maxHeight - scrollHeight) / (barCount - \_barsViewable)

\_scrollBarArea = New Rectangle(Width - 20, 19, 12, maxHeight)

\_scroll = New Rectangle(Width - 20, 19 + (\_scrollPosition \* scrollSpeed), 12, scrollHeight)

\_topPart = New Rectangle(Width - 20, 10, 12, 8)

\_bottomPart = New Rectangle(Width - 20, Height - 10, 12, 8)

Dim colorTopPart As Brush

Dim colorBottomPart As Brush

Dim colorScroll As Brush

If \_mouseOverTopPart = True Then

colorTopPart = Brushes.Black

Else

colorTopPart = Brushes.Gray

End If

If \_mouseOverBottomPart = True Then

colorBottomPart = Brushes.Black

Else

colorBottomPart = Brushes.Gray

End If

If \_mouseOverScrollBar = True Then

colorScroll = New LinearGradientBrush(\_scroll, Color.Bisque, Color.Gray, LinearGradientMode.Horizontal)

Else

colorScroll = New LinearGradientBrush(\_scroll, Color.White, Color.Gray, LinearGradientMode.Horizontal)

End If

grfx.DrawRectangle(Pens.Black, \_topPart)

grfx.FillRectangle(Brushes.LightGray, \_topPart)

grfx.DrawRectangle(Pens.Black, \_bottomPart)

grfx.FillRectangle(Brushes.LightGray, \_bottomPart)

Dim points(2) As PointF

points(0) = New PointF(\_topPart.Left, \_topPart.Bottom - 1)

points(1) = New PointF(\_topPart.Right, \_topPart.Bottom - 1)

points(2) = New PointF((\_topPart.Left + \_topPart.Right) / 2, \_topPart.Top + 1)

grfx.FillPolygon(colorTopPart, points)

points(0) = New PointF(\_bottomPart.Left, \_bottomPart.Top + 1)

points(1) = New PointF(\_bottomPart.Right, \_bottomPart.Top + 1)

points(2) = New PointF((\_bottomPart.Left + \_bottomPart.Right) / 2, \_bottomPart.Bottom - 1)

grfx.FillPolygon(colorBottomPart, points)

grfx.DrawRectangle(Pens.Black, \_scrollBarArea)

grfx.FillRectangle(Brushes.DarkGray, \_scrollBarArea)

grfx.DrawRectangle(Pens.Black, \_scroll)

grfx.FillRectangle(colorScroll, \_scroll)

End Sub

Private WriteOnly Property ScrollPositionY As Integer

Set

Dim barCount As Integer = GetIndexChartBar("QQQWWW")

Dim maxHeight As Integer = Height - 30

Dim scrollHeight As Decimal = (maxHeight / barCount) \* \_barsViewable

Dim scrollSpeed As Decimal = (maxHeight - scrollHeight) / (barCount - \_barsViewable)

Dim index As Integer = 0

Dim distanceFromLastPosition = 9999

While index < barCount

Dim newPositionTemp As Integer = (index \* scrollSpeed) + (scrollHeight / 2) + (30 / 2)

Dim distanceFromCurrentPosition = newPositionTemp - value

If distanceFromLastPosition < 0 Then

If distanceFromCurrentPosition < distanceFromLastPosition Then

\_scrollPosition = index - 1

PaintChart()

Exit Property

End If

Else

If distanceFromCurrentPosition > distanceFromLastPosition Then

\_scrollPosition = index - 1

If \_scrollPosition + \_barsViewable > GetIndexChartBar("QQQWWW") Then

\_scrollPosition = GetIndexChartBar("QQQWWW") - \_barsViewable

End If

PaintChart()

Exit Property

End If

End If

distanceFromLastPosition = distanceFromCurrentPosition

index += 1

End While

End Set

End Property

Public Sub ScrollOneup()

If \_scrollPosition = 0 Then Exit Sub

\_scrollPosition -= 1

PaintChart()

End Sub

Public Sub ScrollOneDown()

If \_scrollPosition + \_barsViewable >= GetIndexChartBar("QQQWWW") Then Exit Sub

\_scrollPosition += 1

PaintChart()

End Sub

Private Sub GanttChart\_Click(sender As System.Object, e As MouseEventArgs) Handles MyBase.MouseClick

If e.Button = Windows.Forms.MouseButtons.Left Then

If \_mouseOverBottomPart = True Then

ScrollOneDown()

ElseIf \_mouseOverTopPart = True Then

ScrollOneup()

End If

End If

End Sub

Private Sub GanttChart\_MouseWheel(sender As Object, e As MouseEventArgs) Handles Me.MouseWheel

If e.Delta > 0 Then

ScrollOneup()

Else

ScrollOneDown()

End If

End Sub

#End Region

#Region "Save"

Public Sub SaveImage(filePath As String)

\_objGraphics.SmoothingMode = SmoothingMode.HighSpeed

\_objGraphics.Clear(BackColor)

If \_headerFromDate = Nothing Or \_headerToDate = Nothing Then Exit Sub

DrawHeader(\_objGraphics, Nothing)

DrawNetHorizontal(\_objGraphics)

DrawNetVertical(\_objGraphics)

DrawBars(\_objGraphics, True)

\_objBmp.Save(filePath)

End Sub

#End Region

Private Enum MouseOverPart

Empty

Bar

BarLeftSide

BarRightSide

End Enum

End Class