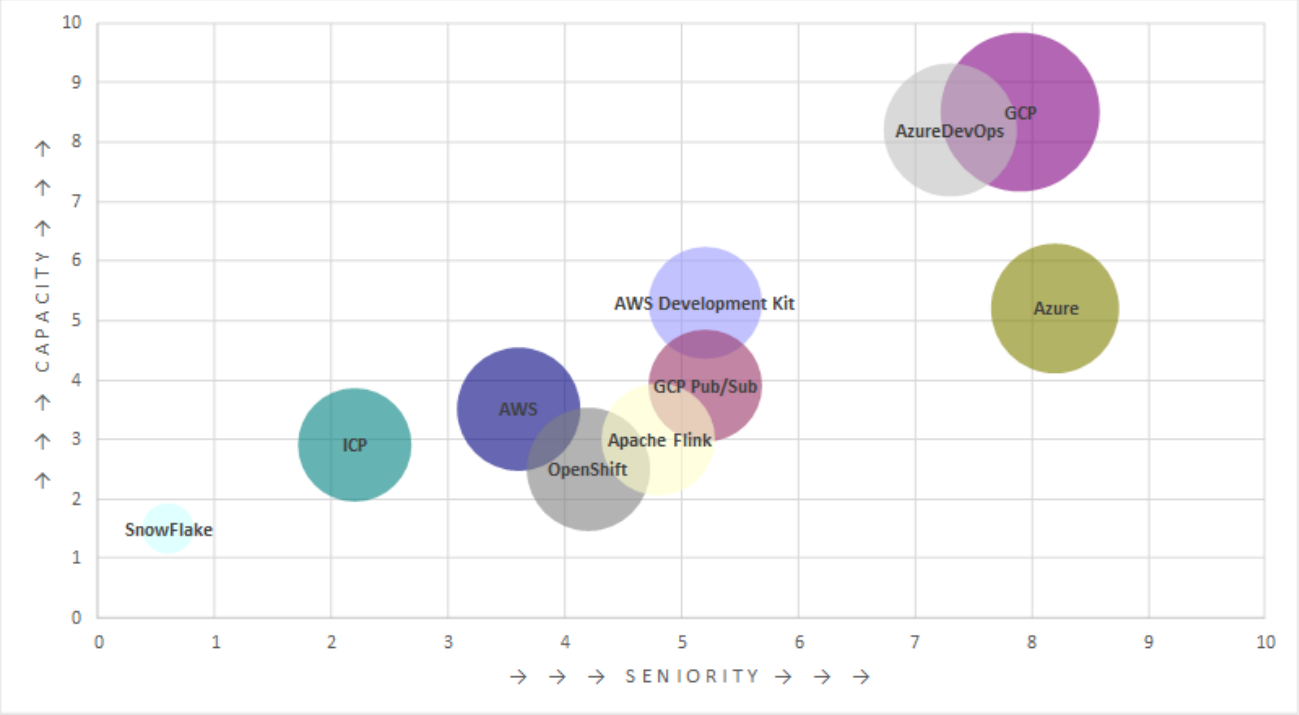
Gráfico de Bolhas com C# e Interoperabilidade com Excel

Exemplo de aplicação **Windows Forms** (**C#**) demonstrando a utilização do *namespace* **Microsoft.Office.Interop.Excel** para criar um gráfico de bolhas (*Bubble Chart*) dinamicamente.



Pré-requisitos:

- Visual Studio Code (Download)
- Microsoft Excel 16.0 Object Library
- .NET Framework 4.7.2

Funcionalidades ilustradas:

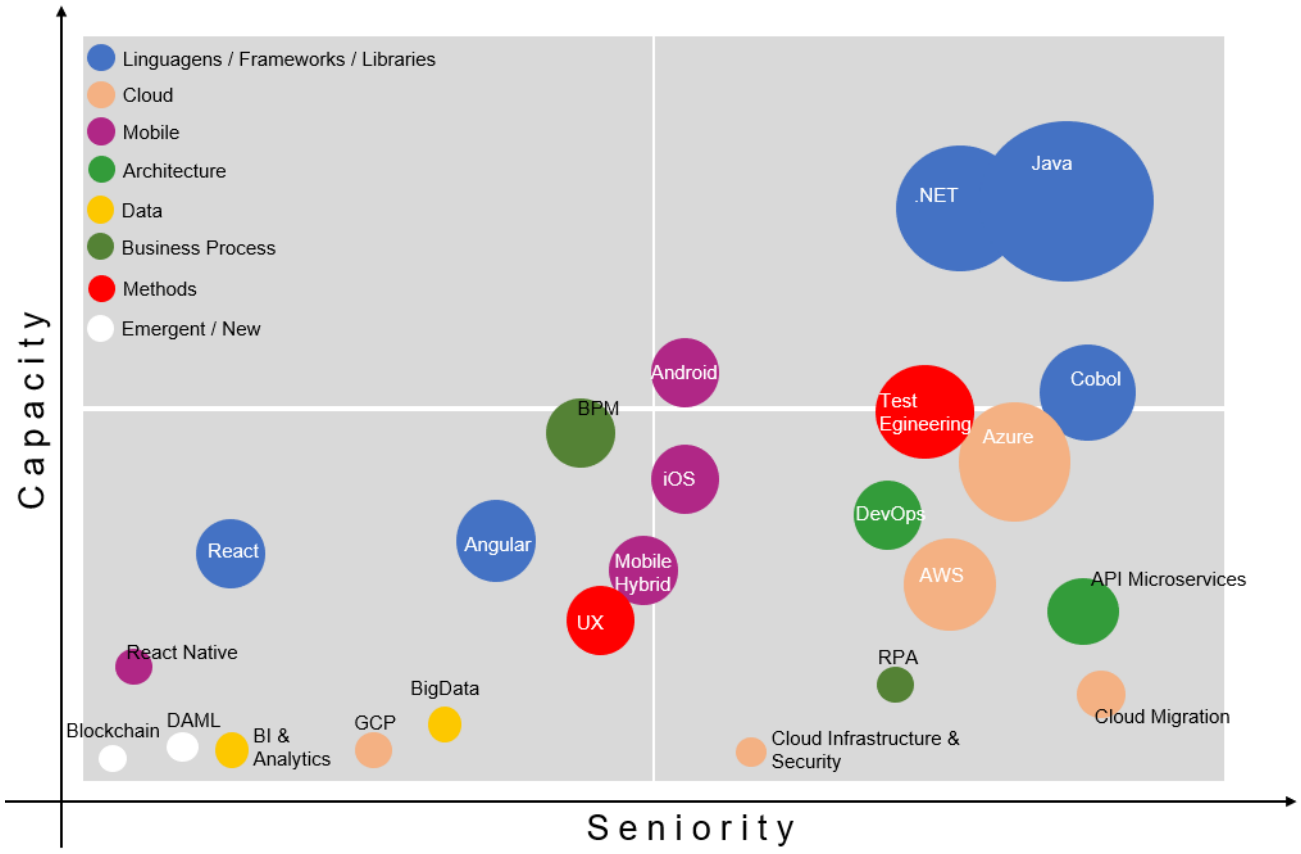
- Instanciar o Microsoft Excel através do Office.Interop
- Adicionar Gráfico de Bolha numa Planilha
- Personalizar os Eixos X e Y do Gráfico
- Personalizar os Labels dos Dados apresentados
- Adicionar Campos Customizados no Gráfico
- Aplicar Estilos no Gráfico
- Aplicar Cores Dinamicamente para os Data Points
- Nomear e Consumir Ranges de Planilhas (Worksheets)
- Capturar/Exportar Gráfico em formato PNG
- Converter uma Range em Lista Genérica
- Salvar Excel Workbook
- Carregar ComboBox e ListBox com objetos Dictionary<string, int>
- Manipular Pastas e Arquivos com System.IO
- Identificar Arquivo Mais Recente numa Determinada Pasta

Motivação

A origem desta aplicação de exemplo está no exercício para apresentar de forma visual e dinâmica, a capacidade e senioridade, de uma determinada empresa/time/grupo de profissionais de tecnologia, representada num modelo de quadrante, seguindo o conceito abaixo:

Alta Capacidade ou Disponibilidade de Profissionais. **Baixa Senioridade.**	**Alta Capacidade ou Disponibilidade de Profissionais.** **Alta Senioridade.**
Baixa Capacidade ou Disponibilidade de Profissionais. **Baixa Senioridade.**	**Baixa Capacidade ou Disponibilidade de Profissionais.** **Alta Senioridade.**

Exemplo desta representação gráfica:

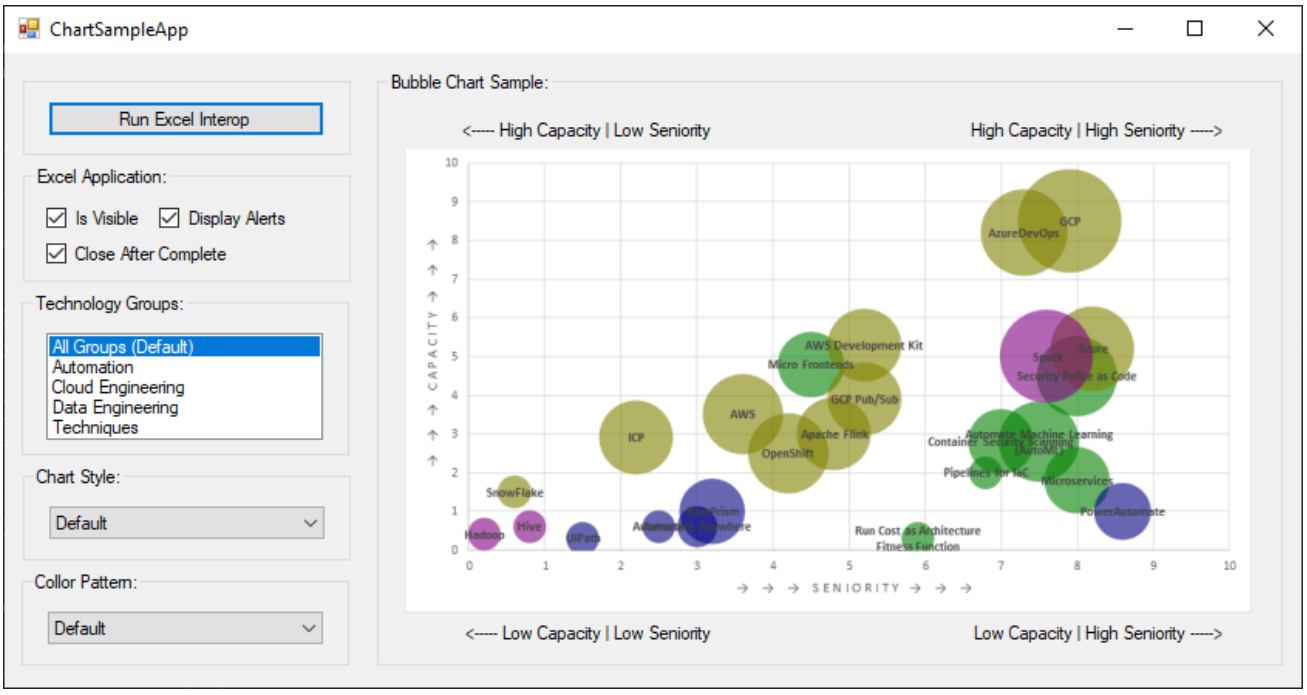


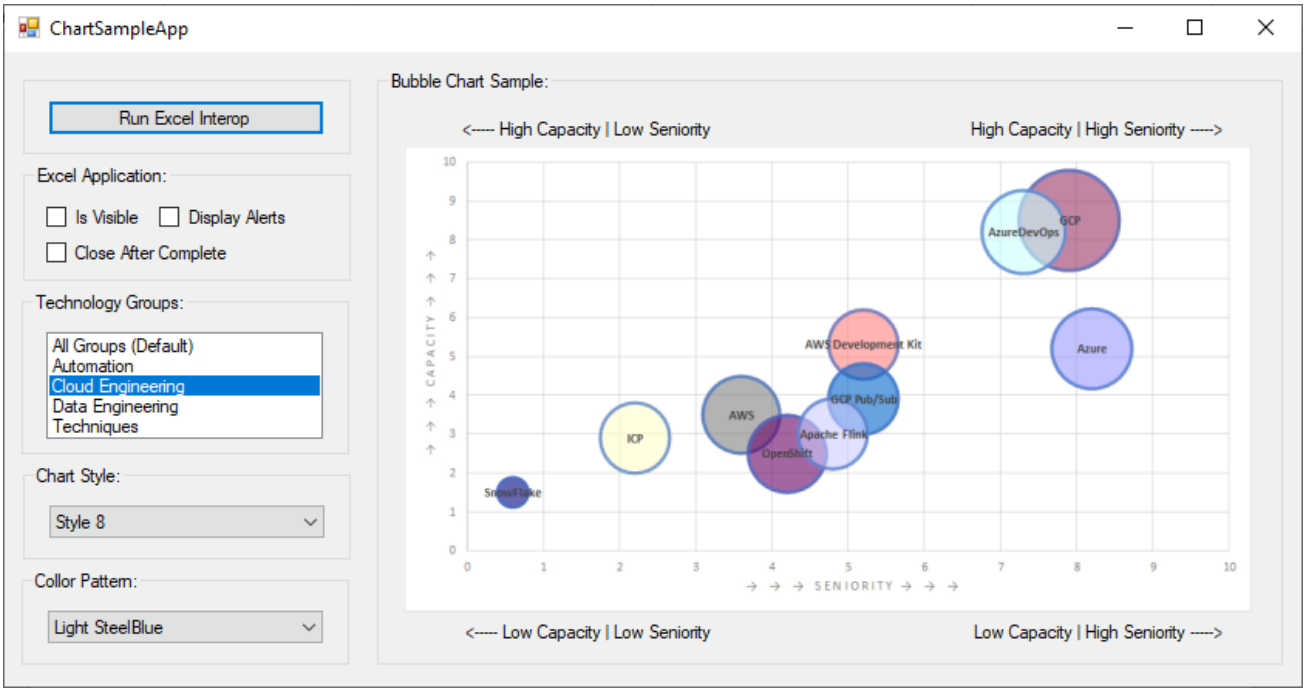
Aplicação de Exemplo

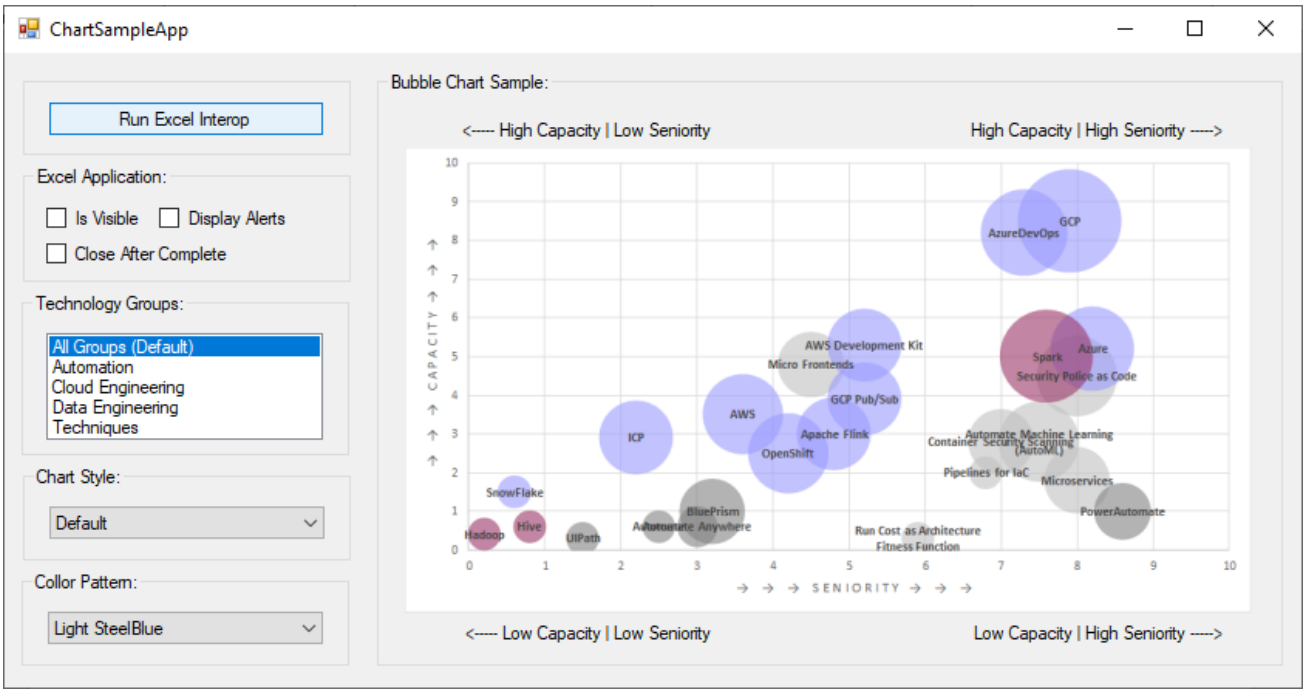
Para ilustrar as funcionalidades listadas acima, utilizei uma aplicação desktop (*Windows Forms*), o que tornou mais simples a implementação da interface e utilização do *Namespace* **Microsoft.Office.Interop**. Esta aplicação de exemplo permite que o usuário escolha se deseja manter a instância do **Excel** visível durante o processamento do gráfico, se deseja encerrar a instância do **Excel** após concluir o processamento, também permite escolher entre gerar o gráfico apresentando os números de todos os grupos de tecnologia, ou apenas um grupo selecionado. A aplicação também oferece a possibilidade de aplicar estilos e cores ao gráfico com base na seleção do usuário.

Ao clicar no botão **[Run Excel Interop]** a aplicação criará uma instância do **Excel**, adicionando um novo **Workbook** e **Worksheets** necessárias para processamento do gráfico, seguindo as opções selecionadas pelo usuário. Uma imagem do gráfico processado será capturada na Área de Transferência (*Clipboard*), e persistida em arquivo no formato **PNG**. Finalmente, a imagem é exibida na aplicação de exemplo, concluindo o processamento. As imagens a seguir foram capturadas em tempo de execução.

Screenshots:







O modelo de dados

Adotei um modelo de dados simples, contendo apenas as informações necessárias para gerar o gráfico de bolhas dinamicamente. A classe **QuadranteModel** implementa esse modelo. E a classe **TechnologyCapacity** implementa o método **GetInMemoryData,** que utiliza uma lista genérica para carregar os dados necessários para gerar o gráfico e ilustrar as funcionalidades deste exemplo.

```
public class QuadrantModel
    {
        public int GroupID { get; set; }
        public string GroupName { get; set; }
```

```
        public string TechnologyName { get; set; }
        public double Seniority { get; set; }
        public double Capacity { get; set; }
        public double BubbleSize { get; set; }
    }
```

O fluxo da aplicação

Direto ao ponto, não vou abordar aqui o fluxo de inicialização da aplicação e a carga dos controles de lista na interface, vamos falar apenas do fluxo necessário para gerar o gráfico dinamicamente, utilizando o *Namespace* **Microsoft.Office.Interop.Excel**. A tabela abaixo a sequência de métodos executados para gerar o gráfico.

1	CreateQuadrantSpreadsheets	Cria as planilhas **[Report]** para armazenar os dados e **[Chart]** onde será adicionado o gráfico de bolhas.
2	LoadQuadrantData	Carrega os dados utilizando o método **GetInMemoryData** da classe **TechnologyCapacity.**
2.1	SetReportWorksheetRanges	Define os objetos **Range** que serão utilizados para gerar o gráfico dinamicamente.
4	GenerateQuadrantChart	Inicia o conjunto de instruções para gerar o gráfico de bolhas com base nas opções do usuário.
4.1	InsertBubbleChart	Adiciona um objeto **Chart** do tipo **xlBubble** utilizando os dados da planilha **[Data].**
4.2	FormatChartDataSeries	Define as propriedades que formatam as séries de dados do novo gráfico.
4.3	FormatChartAxis	Define as propriedades de formatação dos eixos **X** e **Y** do novo gráfico.
4.4	SetDataLabelsValuesFromCells	Define os valores dos **DataLabels** a partir dos valores contidos nas células de um objeto **Range.**
4.5	SetBubbleCollorsByCategory	Define as cores das bolhas de acordo com a categoria ou grupo de tecnologia que ela pertence.
4.6	ApplyChartStyle	Aplica um estilo ao gráfico. Os estilos são representados pelo **Excel** através de números inteiros.
4.7	CopyChartPictureToClipboard	Copia uma imagem do gráfico para *Área de Transferência* utilizando o método **CopyPicture** do objeto **Chart**.
4.8	SaveChartPictureFromClipboard	Salva a imagem da *Área de Transferência* num arquivo com formato **PNG**.
5	SaveWorkbookAndCloseExcel	Salva as alterações realizadas nas planilhas, fecha o **Workbook** e encerra a instância do **Excel**.

Criando as planilhas

O método CreateQuadrantSpreadsheets é responsável por criar as planilhas **[Report]** para armazenar os dados e **[Chart]** onde será adicionado o gráfico de bolhas.

```
private void CreateQuadrantSpreadsheets()
{
    Workbook reportWorkbook = AppExcel.Workbooks.Add();
    Worksheet reportWorksheet = (Worksheet)reportWorkbook.ActiveSheet;
    Worksheet chartWorksheet = (Worksheet)reportWorkbook.Worksheets.Add();

    reportWorksheet.Cells[1, "A"] = "Group Name";
```

```
    reportWorksheet.Cells[1, "B"] = "Technology";
    reportWorksheet.Cells[1, "C"] = "Seniority";
    reportWorksheet.Cells[1, "D"] = "Capacity";
    reportWorksheet.Cells[1, "E"] = "Bubble Size";

    reportWorksheet.Name = "Data";
    chartWorksheet.Name = "Chart";

    RemoveWorksheetsGridLInes();
}
```

Carregando os Dados

O método **LoadQuadrantData** Carrega os dados utilizando o método **GetInMemoryData** da classe **TechnologyCapacity.**

```
private void LoadQuadrantData()
{
    TechnologyCapacity techCapacity = new TechnologyCapacity();
    QuadrantDataList = techCapacity.GetInMemoryData(TechnologyGroupId);

    Worksheet reportWorksheet = (Worksheet)AppExcel.Worksheets["Data"];

    int iCurrentRow = 2;

    foreach (QuadrantModel quadrantItem in QuadrantDataList)
    {
        reportWorksheet.Cells[iCurrentRow, "A"] = quadrantItem.GroupName.ToString();
        reportWorksheet.Cells[iCurrentRow, "B"] = quadrantItem.TechnologyName.ToString();
        reportWorksheet.Cells[iCurrentRow, "C"] =
double.Parse(quadrantItem.Seniority.ToString().Replace(".", ","));
        reportWorksheet.Cells[iCurrentRow, "D"] =
double.Parse(quadrantItem.Capacity.ToString().Replace(".", ","));
        reportWorksheet.Cells[iCurrentRow, "E"] =
double.Parse(quadrantItem.BubbleSize.ToString().Replace(".", ","));
        iCurrentRow++;
     }

     int iFirstRow = 2;
     int iLastRow = iCurrentRow--;
}
```

Nomeando os objetos Range

O método **SetReportWorksheetsRanges** define os objetos **Range** da planilha **[Report]** que serão utilizados para gerar o gráfico dinamicamente.

```
private void SetReportWorksheetRanges(Worksheet reportWorksheet, int iFirstRow, int iLastRow)
{
   Range titleRow = reportWorksheet.get_Range("A1", "E1");
   Range tableRange = reportWorksheet.get_Range("A1", "E" + iLastRow.ToString());
   Range chartDataRange = reportWorksheet.get_Range("C" + iFirstRow.ToString(), "E" +
iLastRow.ToString());
   Range technologyRange = reportWorksheet.get_Range("B" + iFirstRow.ToString(), "B" +
iLastRow.ToString());

   reportWorksheet.Names.Add("TitleRange", titleRow);
   reportWorksheet.Names.Add("TableRange", tableRange);
```

```
    reportWorksheet.Names.Add("ChartDataRange", chartDataRange);
    reportWorksheet.Names.Add("TechnologyRange", technologyRange);

     FormatReportWorksheet(reportWorksheet, titleRow, tableRange);
}
```

Adicionando o Gráfico de Bolhas

O método **InsertBubbleChart** adiciona um objeto **Chart** do tipo **xlBubble** utilizando os dados da planilha **[Data].**

```
private Chart InsertBubbleChart(Worksheet chartWorksheet, Worksheet reportWorksheet)
{
    ChartObjects xlCharts = (ChartObjects)chartWorksheet.ChartObjects(Type.Missing);
    ChartObject myChart = xlCharts.Add(5, 5, 600, 330);

    Chart chartPage = myChart.Chart;

    chartPage.ChartType = XlChartType.xlBubble;
    chartPage.HasLegend = false;

    Range chartRange = reportWorksheet.Range["ChartDataRange"];
    chartPage.SetSourceData(chartRange);
    return chartPage;
}
```

Formatando os DataSeries

O método **FormatChartDataSeries** define as propriedades que formatam as séries de dados do novo gráfico.

```
private Series FormatChartDataSeries(Chart chartPage)
{
    Series series1 = (Series)chartPage.SeriesCollection(1);
    series1.HasDataLabels = true;

    series1.DataLabels(0).Position = XlDataLabelPosition.xlLabelPositionCenter;
    series1.DataLabels(0).ShowCategoryName = false;
    series1.DataLabels(0).ShowValue = false;
    series1.DataLabels(0).ShowRange = true;
    series1.DataLabels(0).Font.Bold = true;
    return series1;
}
```

Formatando os eixos do gráfico

O método **FormatChartAxis** define as propriedades de formatação dos eixos **X** e **Y** do novo gráfico.

```
private void FormatChartAxis(Chart chartPage)
{
    Axis yAxis = (Axis)chartPage.Axes(XlAxisType.xlValue, XlAxisGroup.xlPrimary);
    Axis xAxis = (Axis)chartPage.Axes(XlAxisType.xlCategory, XlAxisGroup.xlPrimary);

    string xAxisTitle = "   →    →    →    S E N I O R I T Y   →    →    →";
    string yAxisTitle = "   →    →    →    C A P A C I T Y   →    →    →";
```

```
    xAxis.HasTitle = true;
    xAxis.AxisTitle.Text = xAxisTitle;
    xAxis.MinimumScale = 0;
    xAxis.MaximumScale = 10;

    yAxis.HasTitle = true;
    yAxis.AxisTitle.Text = yAxisTitle;
    yAxis.MinimumScale = 0;
    yAxis.MaximumScale = 10;
}
```

Definindo os Labels das Bolhas

O método **SetDataLabelsValuesFromCells** define os valores dos **DataLabels** a partir dos valores contidos nas células de um objeto **Range.** Esta opção é conhecida pelos usuários do **Excel** através da página de propriedades do gráfico que permite selecionar esta opção e indicar um conjunto de células que representam os valores para os **Labels**.

```
private void SetDataLabelsValuesFromCells(Worksheet reportWorksheet, Series series1)
{
    int msoChartFieldRange = 7;
    int msoFieldPosition = 0;

    Range technologyRange = reportWorksheet.Range["TechnologyRange"];
    string technologyRangeAbsoluteAddress = "=" + technologyRange.Worksheet.Name + "!" +
technologyRange.Address;

    series1.DataLabels(0).Format.TextFrame2.TextRange.InsertChartField(msoChartFieldRange,
technologyRangeAbsoluteAddress, msoFieldPosition);
}
```

Colorindo as bolhas

O método **SetBubbleCollorsByCategory** define as cores das bolhas de acordo com a categoria ou grupo de tecnologia que ela pertence.

```
private void SetBubbleCollorsByCategory(Series series1)
{
    int iBubbleColor = CollorPatternCode;

    QuadrantModel firstTechnologyGroup = QuadrantDataList.Find(QuadrantDetail =>
QuadrantDetail.TechnologyName == series1.DataLabels(1).Text.ToString());

    string sGroupName = firstTechnologyGroup.GroupName;

    int iChartDataPoints = QuadrantDataList.Count;

    for (int iDataLabel = 1; iDataLabel <= iChartDataPoints; iDataLabel++)
    {
QuadrantModel objQuadrantDetail = QuadrantDataList.Find(QuadrantModel =>
QuadrantModel.TechnologyName == series1.DataLabels(iDataLabel).Text.ToString());

if (objQuadrantDetail.GroupName != sGroupName)
{
    iBubbleColor++;
    sGroupName = objQuadrantDetail.GroupName;
```

```
}

if (TechnologyGroupId > 0)
{
    iBubbleColor++;
}

series1.Points(iDataLabel).Interior.ColorIndex = iBubbleColor;
series1.Points(iDataLabel).Format.Fill.Transparency = 0.4;
    }
}
```

Aplicando um estilo no gráfico

O método **ApplyChartStyle** aplica um estilo ao gráfico. Os estilos são representados pelo **Excel** através de números inteiros. Para obter todos os estilos você pode consultar a documentação do **Excel** ou criar uma macro e analisar o código **VBA**.

```
private void ApplyChartStyle(Chart chartPage)
{
    chartPage.ChartStyle = ChartStyleCode;
}
```

Copiando uma imagem do gráfico

O método **CopyChartPictureToClipboard** copia uma imagem do gráfico para *Área de Transferência* utilizando o método **CopyPicture** do objeto **Chart**.

```
private void CopyChartPictureToClipboard(Worksheet reportWorksheet, Chart chartPage)
{
    System.Windows.Forms.Clipboard.Clear();
    chartPage.CopyPicture();
    reportWorksheet.Paste();
    reportWorksheet.Shapes.Item(reportWorksheet.Shapes.Count).Cut();
}
```

Salvando a imagem do gráfico

O método **SaveChartPictureFromClipboard** recupera a imagem da *Área de Transferência* e salva em arquivo com formato **PNG**.

```
private void SaveChartPictureFromClipboard()
{
    StringBuilder chartImageFilename = new StringBuilder();
    chartImageFilename.Append(Tools.GetImagesFolder());
    chartImageFilename.Append("\\TechnologyQuadrant_");
    chartImageFilename.Append(DateTime.Now.ToString("MMM_ddd_HHmmss").ToUpper());
    chartImageFilename.Append(".png");

    BitmapEncoder bmpEncoder = new BmpBitmapEncoder();

    bmpEncoder.Frames.Add(BitmapFrame.Create(System.Windows.Clipboard.GetImage()));

    using (System.IO.MemoryStream outStream = new System.IO.MemoryStream())
```

```
    {
       bmpEncoder.Save(outStream);
       System.Drawing.Image imgGraph = new System.Drawing.Bitmap(outStream);
       imgGraph.Save(chartImageFilename.ToString());
    }
}
```

Finalizando o trabalho

O método **SaveWorkbookAndCloseExcel** salva as alterações realizadas nas planilhas, fecha o **Workbook** e encerra a instância do **Excel**.

```
private void SaveWorkbookAndCloseExcel()
{
    StringBuilder workbookFilename = new StringBuilder();
    workbookFilename.Append(Tools.GetExcelFolder());
    workbookFilename.Append("\\TechnologyQuadrant_");
    workbookFilename.Append(DateTime.Now.ToString("MMM_ddd_HHmmss").ToUpper());
    workbookFilename.Append(".xlsx");

    if (CloseExcelAfterCompleteFlag || !AppExcel.Visible)
    {
       AppExcel.ActiveWorkbook.SaveAs(workbookFilename.ToString());
       AppExcel.ActiveWorkbook.Close();
       AppExcel.Quit();
       AppExcel = null;
    }
}
```

Conclusão

Esse exemplo ilustra apenas uma das formas que você pode utilizar para gerar um gráfico dinamicamente utilizando os recursos de interoperabilidade do **Microsoft Office**. Você pode consultar a documentação oficial disponível neste link https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/. A aplicação de exemplo e o código-fonte está disponível neste repositório: https://github.com/cmattos/BubbleChartSample/.