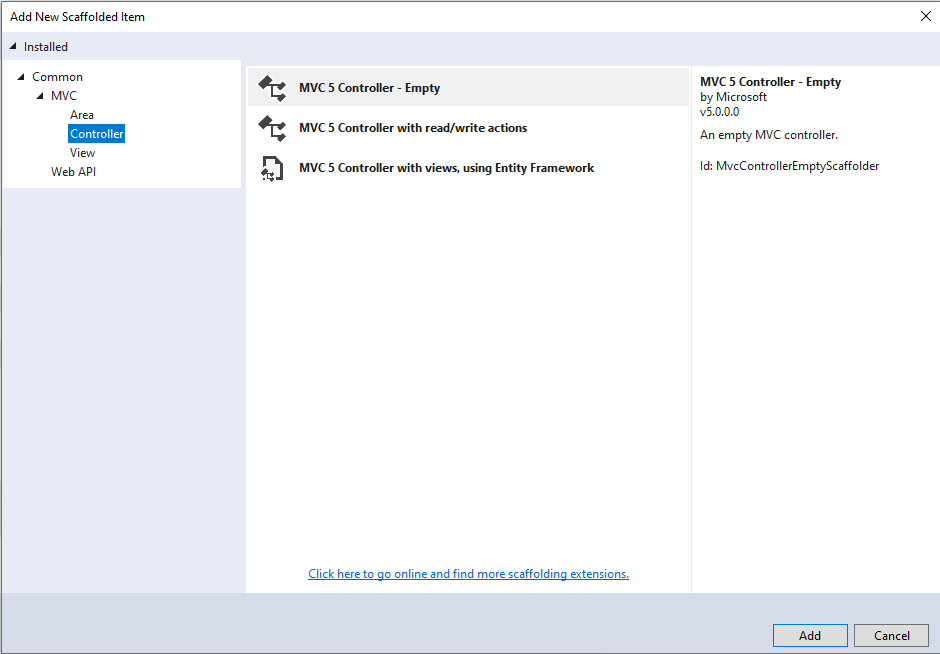
Na Invoice.cshtml mamy przycisk GenerujPdf

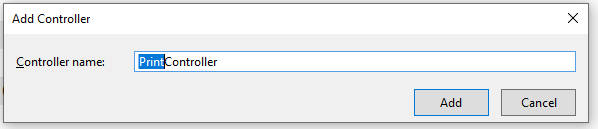
|  |
| --- |
| <button class="btn btn-secondary mb-3"  onclick="generatePdf()">  Generuj PDF  </button> |

Do którego podpięta jest funkcja generatePdf()

Musimy stworzyć nowy kontroler i akcje odpowiedzialne za generowanie Pdf.

Klikamy w projekcie na katalog Controllers, wybieramy Add i potem Controller.





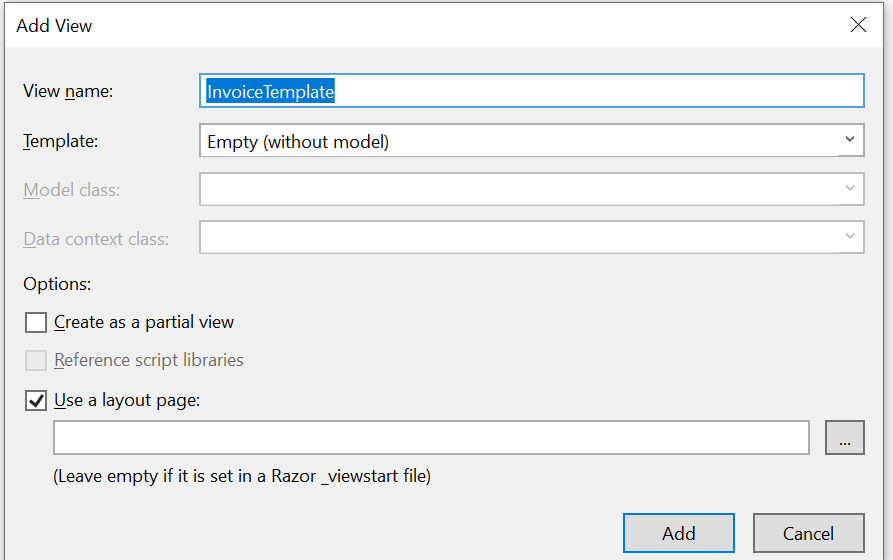
Został wygenerowany plik

|  |
| --- |
| namespace InvoiceManager.Controllers  {  public class PrintController : Controller  {  // GET: Print  public ActionResult Index()  {  return View();  }  }  } |

A w widokach został utworzony katalog Print

Wykorzystamy przygotowany wcześniej szablon faktury

W Content tworzymy folder Images I wgrywamy tam logo.  
W Content tworzymy też nowy plik css i wklejamy do niego zawartość z css, który przygotowaliśmy do faktury.  
Tworzymy również nowy widok w katalogu Print: InvoiceTemplate



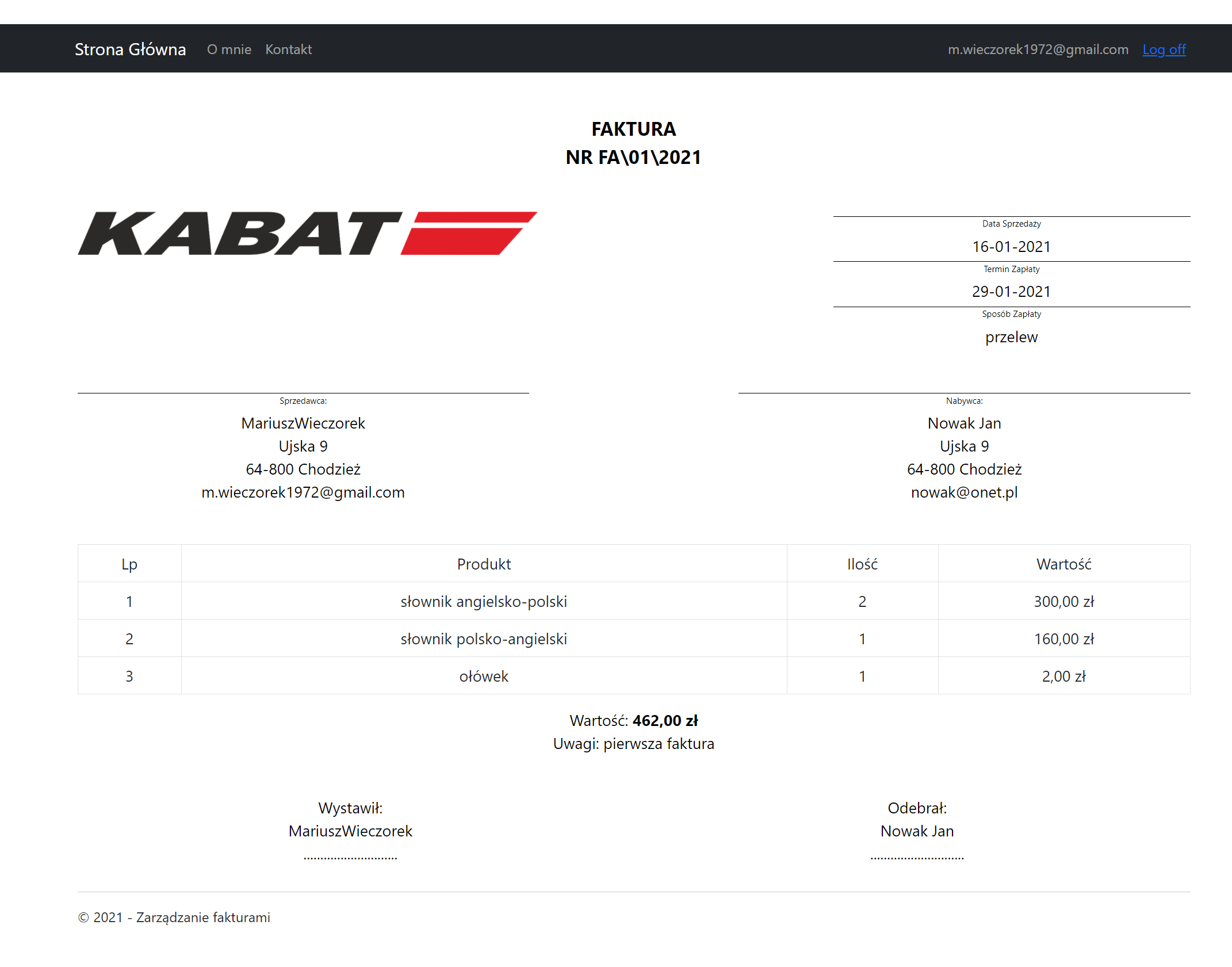
Do niego kopiujemy to co mamy wewnątrz sekcji <body></body>

|  |
| --- |
| @model InvoiceManager.Models.Domains.Invoice  @{  ViewBag.Title = "InvoiceTemplate";  }  <link href="@Url.Content("~/Content/PrintInvoice.css")" rel="stylesheet" type="text/css"/>  <div class="invoice-font">  <div class="row">  <div class="col-12 header-font">  <div>Faktura</div>  <div>Nr @Model.Title</div>  </div>  </div>  <div class="row mt-5">  <div class="col-5">  <img src="@Url.Content("~/Content/Images/logo\_kabat.png")" alt="logo" class="logo" />  </div>  <div class="col-4 offset-3">  <div class="top-line">  <div class="top-small-caption">  Data Sprzedaży  </div>  <div>  @Model.CreatedDate.ToString("dd-MM-yyyy")  </div>  </div>  <div class="top-line">  <div class="top-small-caption">  Termin Zapłaty  </div>  <div>  @Model.PaymentDate.ToString("dd-MM-yyyy")  </div>  </div>  <div class="top-line">  <div class="top-small-caption">  Sposób Zapłaty  </div>  <div>  @Model.MethodOfPayment  </div>  </div>  </div>  </div>  <div class="row mt-5">  <div class="col-5">  <div class="top-line">  <div class="top-small-caption">  Sprzedawca:  </div>  <div>@Model.User.Name</div>  <div>@Model.User.Address.Street @Model.User.Address.Number</div>  <div>@Model.User.Address.PostalCode @Model.User.Address.City</div>  <div>@Model.User.Email</div>  </div>  </div>  <div class="col-5 offset-2">  <div class="top-line">  <div class="top-small-caption">  Nabywca:  </div>  <div>@Model.Client.Name</div>  <div>@Model.Client.Address.Street @Model.Client.Address.Number</div>  <div>@Model.Client.Address.PostalCode @Model.Client.Address.City</div>  <div>@Model.Client.Email</div>  </div>  </div>  </div>  <table class="table table-bordered mt-5">  <thead>  <tr>  <th>Lp</th>  <th>Produkt</th>  <th>Ilość</th>  <th>Wartość</th>  </tr>  </thead>  <tbody>  @foreach (var position in Model.InvoicePositions)  {  <tr>  <td>@position.Lp</td>  <td>@position.Product.Name</td>  <td>@position.Quantity</td>  <td>@position.Value zł</td>  </tr>  }    </tbody>  </table>  <div class="row">  <div class="col-12">  <div class="text-right">Wartość: <strong>@Model.Value zł</strong></div>  @if (!string.IsNullOrWhiteSpace(Model.Comments))  {  <div class="text-left">Uwagi: @Model.Comments </div>  }  </div>  </div>  <div class="row mt-5">  <div class="col-6">  <div>Wystawił:</div>  <div>@Model.User.Name</div>  <div>............................</div>  </div>  <div class="col-6">  <div>Odebrał:</div>  <div>@Model.Client.Name</div>  <div>............................</div>  </div>  </div>  </div> |

Tworzymy akcję w kontrolerze Print aby przetestować, czy nasza faktura poprawnie się wyświetla.

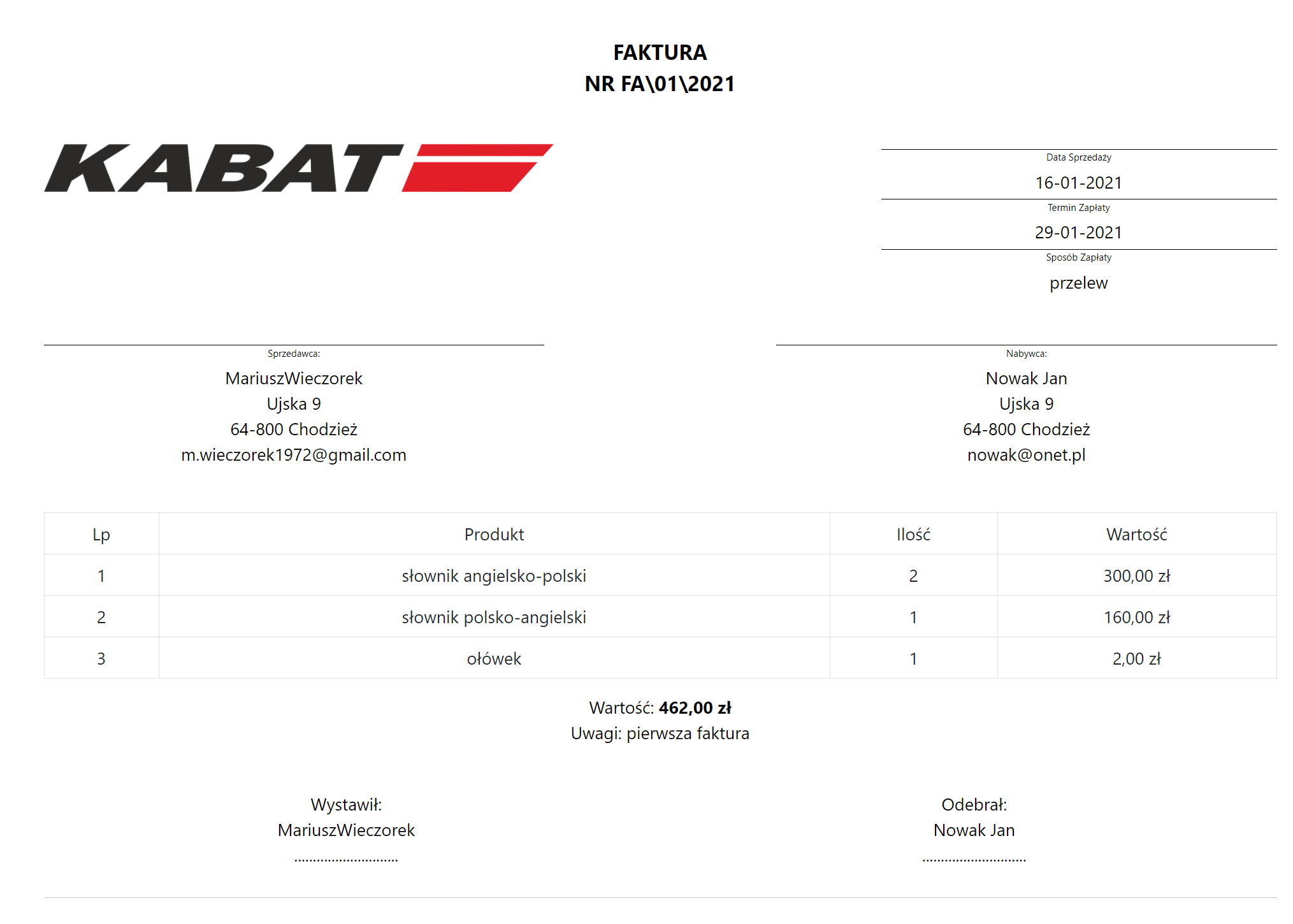
|  |
| --- |
| namespace InvoiceManager.Controllers  {  [Authorize]  public class PrintController : Controller  {  private InvoiceRepository \_invoiceRepository = new InvoiceRepository();  public ActionResult InvoiceTemplate(int id)  {  var userId = User.Identity.GetUserId();  var invoice = \_invoiceRepository.GetInvoice(id,userId);  return View(invoice);  }  }  } |

Strona po uruchomieniu wygląda jak poniżej.



Aby pozbyć się nagłówka i stopki wpisujemy w pliku css powiązanym z fakturą:  
słowo !important powoduje, że nie nadpisze tego inny styl, chyba, że też będzie miał słowo !important.

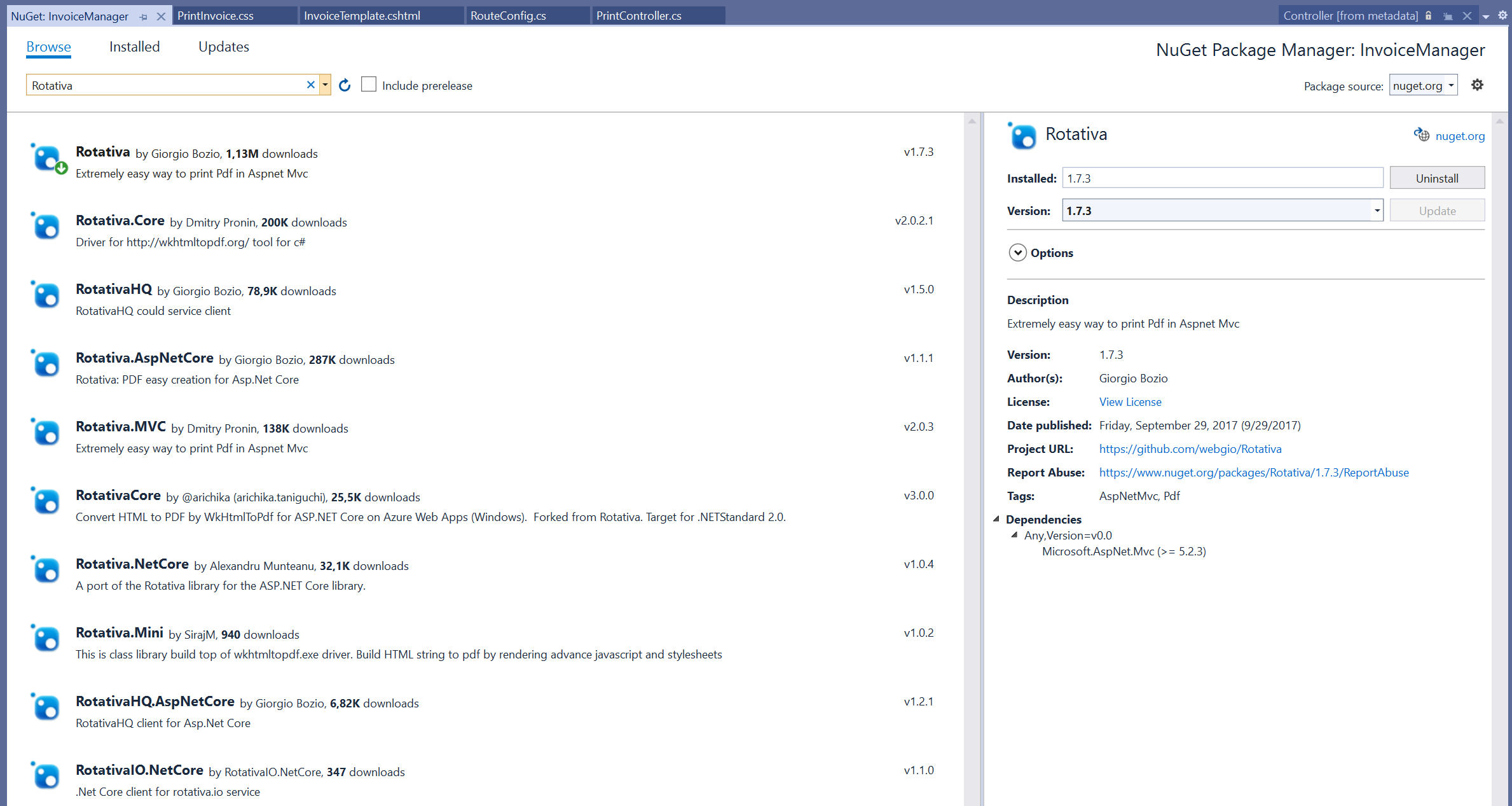
|  |
| --- |
| @page{  size: auto;  margin-top: 20px;  margin-bottom: 20px;  }  nav, footer{  display: none !important;  } |



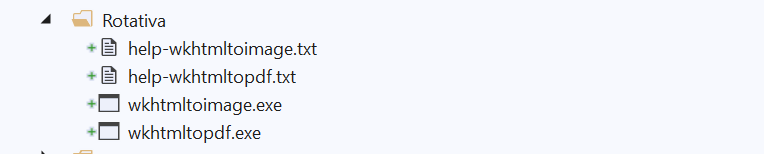
Tak przygotowanego html możemy już generować do PDF’a.

**Dodawanie obsługi generowania plików PDF**

Przechodzimy do zarządzania pakietami, prawy przycisk na projekcie i Manage Nuget Packages  
i wybieramy pakiet o nazwie **Rotativa**



Do projektu został dodany nowy folder



Modyfikujemy plik PrintController.cs  
Uwaga na dołączenie odpowiednich usingów.

|  |
| --- |
| using InvoiceManager.Models.Domains;  using InvoiceManager.Models.Repositories;  using Microsoft.AspNet.Identity;  using Rotativa;  using Rotativa.Options;  using System;  using System.Web.Mvc;  namespace InvoiceManager.Controllers  {  [Authorize]  public class PrintController : Controller  {  private InvoiceRepository \_invoiceRepository = new InvoiceRepository();  public ActionResult InvoiceTemplate(int id)  {  var userId = User.Identity.GetUserId();  var invoice = \_invoiceRepository.GetInvoice(id,userId);  return View(invoice);  }  public ActionResult InvoiceToPdf(int id)  {  var handle = Guid.NewGuid().ToString();  var userId = User.Identity.GetUserId();  var invoice = \_invoiceRepository.GetInvoice(id, userId);  TempData[handle] = GetPdfContent(invoice);  return Json(new  {  FileGuid = handle,  FileName = $@"Faktura\_{invoice.Id}.pdf"  });  }  private byte[] GetPdfContent(Invoice invoice)  {  var pdfResult = new ViewAsPdf(@"InvoiceTemplate", invoice)  {  PageSize = Size.A4,  PageOrientation = Orientation.Portrait  };  return pdfResult.BuildFile(ControllerContext);  }  public ActionResult DownloadInvoicePdf(string fileGuid, string fileName)  {  if (TempData[fileGuid] == null)  throw new Exception("Błąd przy próbie eksportu faktury do PDF.");  var data = TempData[fileGuid] as byte[];  return File(data, "application/pdf", fileName);  }  }  } |

Najpierw musi być wywołana akcja InvoiceToPdf  
Generujemy losowy ciąg znaków: Guid.NewGuid().ToString();  
Pobieramy użytkownika i fakturę ,  
konwertujemy widok do tablicy bajtów TempData[handle] = GetPdfContent(invoice);.   
Do tego używamy rotativy .  
Tablicę bajtów przechowujemy po stronie serwera, z odpowiednim kluczem [handle]   
Akcja zwraca nam Json’a z wygenerowanym kluczem oraz nazwą pliku pdf.

Drugą akcją, jaką będziemy musieli wywołać to Invoice to PDF.

|  |
| --- |
| @section scripts  {  <script type="text/javascript">  // Generowanie PDF'a  function generatePdf() {  if (!confirm("Czy generujemy PDF ?"))  return;  // --------------------------------------------  $.ajax({  type: 'POST',  url: "@Url.Action("InvoiceToPdf", "Print")",  data: {  id: positionId,  invoiceId: '@Model.Invoice.Id'  },  success: function (data) {  if (data.Success) {  // usuwanie z tabeli w html'u  window.location = '@Url.Action("DownloadInvoicePdf", "Print")'  + 'fileGuid=' + data.FileGuid + '&fileName=' + data.FileName;  }  else {  alert(data.Message);  }  },  error: function (data) {  alert(data.Message);  },  dataType: 'json',  cache: false  });  // --------------------------------------------  } |

Jak nam się uda wygenerować PDF’a, to Ajax przekieruje nas do akcji DownloadInvoicePdf.

Kilka niezbędnych wpisów w css aby wydruk dobrze wyglądał

|  |
| --- |
| @page{  size: auto;  margin-top: 20px;  margin-bottom: 20px;  }  nav, footer{  display: none !important;  }  .row {  display: -webkit-box;  display: -webkit-flex;  display: flex !important;  }  .row > div{  -webkit-box-flex: 1;  -webkit-flex: 1;  flex: 1;  }  tr, td, th, thead, thead tr th, tfoot {  page-break-inside: avoid !important;  page-break-before: auto !important;  page-break-after: auto !important;  } |

